
INPUT
Publicación práctica para usuarios de MSX
Revista mensual 1986
Año 1- Número 5 Precio 350 Ptas.
MSX
Domina los sprites
Asómate a las ventanas
Todo entorno al código ASCII

¿SOPORTARAS LAS BARRABASADAS DE ESTE PEQUEÑO GAMBERRO?

JACK THE NIPPER...

MAGNIFICO DIA PARA IR DE COMPRAS

Y portate bien

Por supuesto mama

YA ESTAMOS AQUI

JUS' MICRO

NOW IN...

JACK the NIPPER!

HA NACIDO UNA ESTRELLA

¡OH NO!... EL OTRA VEZ

¡Oye! cárgame Jack the Nipper... que si no...

MONTYS RUNNING OFF!

JACK THE NIPPER

COMPRA ESTE JUEGO... QUE SI NO...

ERBE Software

DE ESTO VA LA COSA

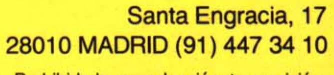
JACK THE NIPPER

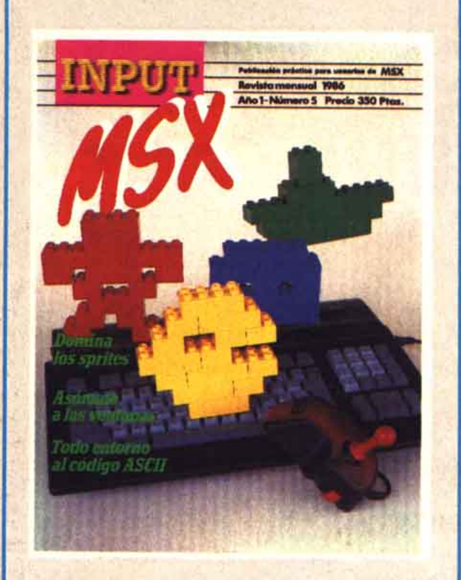
GREMLIN



Es un pequeño malvado. ¿Podrás detenerle? Jack the Nipper gasta las bromas que a todos nos hubiera gustado hacer. Esta es tu ocasión.

Santa Engracia, 17
28010 MADRID (91) 447 34 10





AÑO 1 NUMERO 5

DIRECTOR:
Alejandro Diges
DIRECTOR TECNICO:
Roberto Menéndez
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, Benito Román, Esther de la Cal, Ernesto del Valle, Equipo Molisoft, Javier Portillo.

INPUT MSX es una publicación juvenil de EDICIONES FORUM

GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ Gral. Varela, 35, 3.º-11
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: B-21953-1986

SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13

DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

Se ha solicitado el control OJD

INPUT MSX es independiente y no está vinculada a los distribuidores del estándar.

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas.

INPUT MSX

SUMARIO

MSX EN LA BRECHA

Felicitamos desde aquí a las dos hermanas mayores **INPUT Sinclair** e **INPUT Commodore,** porque cumplen su primer año de presencia en el quiosco.

Hemos recibido unas cuantas cartas de congratulación por el número especial de verano. De cualquier modo sabíamos que un ejemplar de tales características agrada a los usuarios. La experiencia obtenida con un cuadernillo de programas publicado el pasado diciembre en los otros dos **INPUT** así nos lo hacían sospechar.

Este número vuelve a seguir la misma línea de pretensión didáctica marcada desde el principio, aunque no rechazamos publicar listados de programas de vez en cuando. Hay que tener en cuenta que pese a la juventud de la presencia de los sistemas **MSX** en nuestro país, los programas que nos enviáis son en buena proporción de gran calidad. Sabéis que existen agoreros que pronostican un triste futuro a los sistemas **MSX**. Pues bien, en recientes declaraciones a una revista británica de gran relieve europeo, la principal casa de producción y distribución en el mercado francés tiene un buen y amplio equipo de programadores trabajando con ellos. Pero lo que realmente hay que resaltar es que predicen un inmejorable futuro para el **MSX II** (en realidad afirman que este es el futuro), si bien reconocen que el británico es un mercado duro de pelar por la mayor implantación de otras marcas y modelos en él. Sea como fuere, esta casa, **Infogrames,** seguirá lanzando títulos para ampliar la oferta de programas **MSX.** Nuevamente os invitamos a que sigáis enviando vuestras colaboraciones y programas. Con toda seguridad sóis muchos quienes habréis desarrollado unos trabajos excelentes durante las vacaciones de verano.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Alberto Alcocer, 46 - 4.º B. 28016 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido ____________________

Segundo título elegido ____________________

Tercer título elegido ____________________

Programa que te gustaría conseguir ____________________

Qué ordenador tienes ____________________

Nombre ____________________

1.er Apellido ____________________

2.º Apellido ____________________

Fecha de nacimiento ____________________

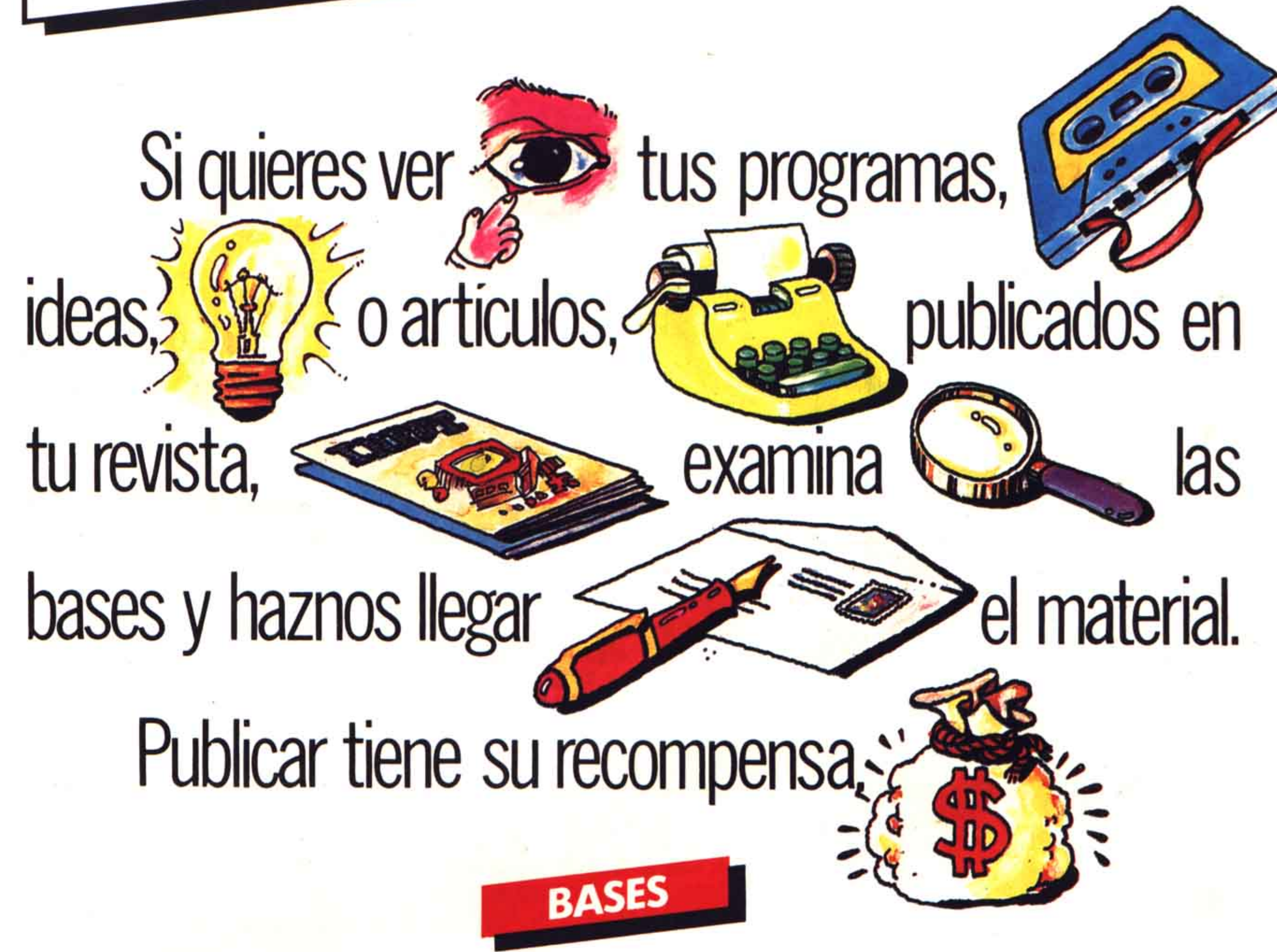
Teléfono ____________________

Dirección ____________________

Localidad ____________________

Provincia ____________________

¡PARTICIPA EN INPUT!



BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas; es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**
  en material microinformático a elegir por los afortunados.

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT MSX

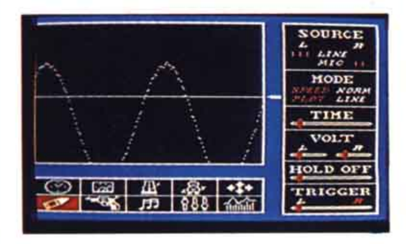
Alberto Alcocer, 46, 4.° B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.



UN OSCILOSCOPIO MSX

Se trata de un cartucho, de la firma NEOS, que se conecta a cualquier MSX y lo convierte en un osciloscopio, presentando en la pantalla la traza de las señales eléctricas a analizar.
Todos los parámetros de control para la presentación (nivel de disparo, periodo de la base de tiempos, sensibilidad del canal, etc) se pueden ajustar a voluntad desde el teclado.
La señal, en la banda de audio, entra en el cartucho a través de un micrófono incorporado.
El sistema, que trabaja en tiempo real, ofrece toda una serie de posibilidades, entre ellas las de congelar la imagen de la señal, almacenarla, etc.
Aunque por el momento nadie se ha decidido, hay varias firmas interesadas en la importación de este sensacional producto.





LOS NUEVOS MITSUBISHI



El catálogo de equipos de la segunda generación, MSX2, cuenta con dos nuevos modelos. Son los Mitsubishi ML-G1 y ML-G3, que fueron presentados en sociedad durante el último Informat (en Barcelona). El modelo ML-G1, más pequeno,incluye en la misma carcasa el teclado y la unidad central. Por sus dimensiones recuerda a los equipos de la primera generación, pero se trata de un potente y completo MSX2, con las siguientes características: 64K de memoria RAM libre para el usuario, 48K de ROM con el sistema operativo y la versión BASIC segunda generación, 128K de VRAM destinados al uso exclusivo del procesador de vídeo y por último otros 32K de ROM con el programa Art Paper, un estupendo programa de diseno gráfico que, a menos que el usuario decida lo contrario, entra en acción nada más encender el equipo.

El modelo ML-G3, el mayor de los dos, tiene un aspecto y unas características más profesionales. En una carcasa de color negro incluye la unidad central y una unidad de *diskettes* de 3.5 pulgadas y 720K de capacidad. El teclado, idéntico al del modelo ML-G1 (con teclado numérico aparte y con un tacto sensacional) va separado de la unidad central.
Las características de memoria son las mismas que las del ML-G1, no ocurre así con los programas incluidos, que para el ML-G3 son los siguientes: un tratamiento de textos,una hoja de cálculo, una base de datos, un programa de gráficos y otro de comunicaciones, que hace uso del *interface* serie RS232 incluido (uno de los aspectos más interesantes de la máquina que dará mucho juego en la previsible era de las comunicaciones entre micros domésticos).
El precio de los equipos (IVA incluido) se situa entorno a las 89.500 (ML-G1) y 165.000 (ML-G3).





SOFT PARA LA EDUCACION

Representantes japoneses de Sony estuvieron visitando recientemente las instalaciones barcelonesas de la empresa Idealogic, interesándose por el *software* que, para el mercado nacional de MSX, esta desarrollando la compania.
Como plato fuerte Idealogic presentó su nueva versión del lenguaje LOGO, más potente y completa, segun se comenta, que las versiones actuales. Esta nueva versión se comercializará a partir de Septiembre, probablemente en cartucho ROM, con el nombre de LOGO MSX Sony.
Otro de los proyectos *software* desarrollados por la empresa, que será presentado durante los próximos meses, es un curso de robótica aplicada a la educación.



1M PARA EL SVI-738

Spectravideo Espana, tiene intención de comercializar, a partir de este mes de Septiembre, una nueva versión de su popular SVI-738 X'Press. Este modelo, de los más completos de la primera generación MSX (entre sus características figuran una unidad de *diskettes* integrada, 80 columnas y un *interface* RS232) incorporará una nueva unidad de *diskettes* que duplicará la capacidad de almacenamiento (actualmente de 360K) situándola en 1 Mbyte (720K formateados) en doble cara.

EL BUZON DE INPUT

Me gustaría saber si se pueden hacer copias de seguridad de cartucho a cartucho o de cartucho a *cassette* y viceversa, y de que manera puedo hacerlo. También me gustaría saber si existe algún juego de baloncesto para MSX. **Si no lo hay, si hay perspectivas de que aparezca alguno. Por último quisiera saber donde puedo conseguir un catálogo con todos los juegos existentes para** MSX.

Fco. David Bellot Moratalla
Alicante

Hacer copias de seguridad de un cartucho a cassette *o disco no es algo inmediato ni mucho menos. Habría que entrar en modificaciones del sistema operativo, en la parte que se encarga de comprobar si hay cartuchos conectados y de transferir el control a los mismos. Otra posibilidad sería la de utilizar alguna placa* hardware *que actuará de* interface *entre el cartucho y el ordenador. No tenemos noticias de que se comercialice ninguna placa de este tipo.*

*Por otro lado y de momento, no hay versiones de baloncesto para **MSX**. Tampoco hay, aunque sabemos que se está preparando, un catálogo oficial y actualizado de* software ***MSX**. Lo único que puedes hacer es ponerte en contacto con los distintos fabricantes e improtadores, que te darán catálogos parciales de sus productos.*

En el número tres de Input MSX **aparecía publicada en la sección Programación de Juegos la instrucción:**

```
PRINT VAL(&B numero binario)
```

para pasar un número binario a decimal. Esta instrucción está equivocada pues la correcta es:

```
PRINT STR$(&B numero
binario)
```

También me gustaría que me explicaséis por qué al comprarme el Knight Lore **e intentar cargarlo en mi** Philips VG-8020 **no salía. Estuve en la tienda donde lo compré pero tras probar varias cintas del mismo juego en el mismo ordenador, ninguna salía, por lo que tuve que llevarme otro juego.**

Ignacio J. Frijeiro Sabater
Málaga

Tienes razón Ignacio. La instrucción está equivocada. Por alguna extraña razón omitimos las comillas de apertura y cierre. La forma correcta de la instrucción es:

```
PRINT VAL("&B numero
binario")
```

Sin embargo no es la única. La que tu mencionas es correcta y también lo es la siguiente, útil para trabajar en modo directo:

```
PRINT &B numero binario
```

*En cuanto a los problemas de carga del **Knight Lore**, es probable que se deban a un mal ajuste del cabezal del* cassette. *Al recibir tu carta hicimos la prueba en la redacción, con un **VG-8020**, y el programa cargó sin problemas a la primera.*



Hola amigos. Os envío una idea curiosa desde mi punto de vista. Se trata de que exite un forma de hacer que los gráficos de un programa aparezcan de golpe y no poco a poco, a medida que se dibujan, como se suele ver. Para ello nos trasladaremos a dos de las rutinas internas de la ROM del ordenador.

La primera de ellas, en la dirección &H41, desactiva la pantalla. La segunda, en la dirección &H44, activa la pantalla.

Con esta dos rutinas no es difícil imaginarse lo que hay que hacer. Tecleando este programa ejemplo se ve mejor:




```
10 CLS:R=RND(-TIME):SCREEN2
20 DEF USR0=&H41:U=USR0(0)
30 '
32 FOR J=1 TO 50
34 EX=170*RND(1)
   :EY=170*RND(1)
36 LINE (100,100)-(EX,EY)
38 NEXT J
39 '
40 DEF USR0=&H44:U=USR0(0)
50 GOTO 50
```

Desde la línea 21 a la 39 (o entre cualesquiera otras si utilizamos otra numeración) se colocarán las instrucciones para los gráficos que deseemos. El ordenador mantiene la pantalla desactivada hasta la línea 40.

Juan Carlos Orós Cabello
Tarragona

Efectivamente, como éstas hay muchas rutinas interesantes en la ROM que podremos utilizar en nuestros programas y que nos ahorrarán bastante trabajo. Os iremos hablando de ellas. Por el momento podéis ir abriendo la boca con las que comenta Juan Carlos.



Quisiera saber si a mi Sony HB-75B **podría convertirlo en un** MSX2. **Me interesaría cualquier información.**

José M. López de Uralde
Zaragoza

EL BUZON DE INPUT

¿Tienen pensado los fabricantes lanzar al mercado algún cartucho o dispositivo para igualar en potencia (sobre todo en vídeo) los MSX **de la primera con los de la segunda generación?**

Jose M. Barrios Navas
Barcelona

El tema de la conversión de los ***MSX*** *de la primera generación en máquinas de la segunda, lo que parece despertar bastante interés (estas dos cartas son sólo una muestra de las recibidas al respecto), es un problema por el momento sin solución. Ninguno de los fabricantes parece dispuesto a embarcarse en la aventura de lanzar un cartucho,* interface *o como queráis llamarlo, para hacer la conversión. Y no es de extrañar si pensamos por un momento como tendría que ser uno de estos* interfaces. *En primer lugar tendría que incluir el nuevo* chip *de vídeo de* ***Yamaha V9938*** *en sustitución del* ***TMS 9918A*** *de la primera generación. Además debería incorporar un mínimo de 64K de VRAM, tal y como especifica la segunda versión del estándar. Pero, además, tendría que incluir la ROM de la segunda generación o al menos una versión reducida de la misma, con los comandos y rutinas para manejar el nuevo procesador de vídeo. Todos estos elementos resultan costosos y acoplarlos adecuadamente representa un costo adicional, por lo que el precio del* interface *resultaría muy elevado y sería difícil rentabilizar el costo de su desarrollo.*

En primer lugar felicitaros por vuestra revista. Me gustaría saber cómo realizar el FLASH en mi Toshiba HX-10**, pues no he encontrado ningún libro o revista que lo explique.**

Miguel Angel García Martínez
Alicante

Gracias por las felicitaciones.

Te vamos a dar una pequeña rutina en código máquina, es la siguiente:

```
;****************************
;INVERSION CARAC VRAM
;INVIERTE MAYUSCULAS EN LA
;TABLA GENERADORA (2048-4096)
;DE LA VRAM
;****************************
               ORG  0D000H
               LOAD 0D000H
;
RVRAM:         EQU  4AH
WVRAM:         EQU  4DH
;
               LD   HL,2568
               LD   E,208
BU1:           CALL RVRAM
               XOR  0FFH
               CALL WVRAM
               INC  HL
               DEC  E
               JR   NZ,BU1
               RET
               END
```

lo que hace es acudir a la tabla generadora de patrones de la VRAM (en el modo 0 de pantalla) invirtiendo los patrones de todas las letras mayúsculas. Si entiendes un poco de código máquina, te diremos que los patrones de las letras mayúsculas empiezan en el byte 2568, que es el valor que cargamos en el registro HL. Si quieres invertir las minúsculas no tienes más que cambiar este valor por 2824, que es el primer byte de los patrones de las minúsculas. La rutina entra en un bucle en el que se dedica a invertir (mediante un XOR) todos los patrones de las mayúsculas. De este modo las mayúsculas aparecen en inverso. La rutina es reubicable, es decir, la puedes cargar en cualquier zona de RAM y seguirá funcionando.

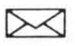
Para ahorrarte trabajo, o por si no andas muy fuerte en código máquina, lo que sigue es un cargador BASIC para la rutina. Tecléalo procurando no equivocarte en ninguno de los valores DATA y escribe RUN. Verás un precioso efecto FLASH que afecta a todas las mayúsculas que aparecen en la pantalla:

```
10 CLEAR 200,&HD000:GOSUB
   500
20 CLS:LOCATE 9,0:PRINT
   "EFECTO FLASH"
30 LOCATE 2,7:PRINT
   "Solo actua sobre las
   MAYUSCULAS"
40 U=USR0(0):FOR J=1 TO 100
   :NEXT:U=USR0(0):FOR J=1
   TO 100:NEXT:GOTO40
500 DEF USR0=&HD000
505 FOR N=&HD000 TO &HD011
510 READ A$:POKE N,VAL
    ("&H"+A$)
520 NEXT
530 DATA 21,08,0A,1E,D0,CD
    ,4A ,00
540 DATA EE,FF,CD,4D,00,23
    ,1D ,20
545 DATA F4,C9
550 RETURN
```

La rutina propiamente dicha, se carga con la subrutina de las líneas 500 a 550. Estas, y la línea 10, es necesario que las incluyas en el programa que utilice la rutina. Las líneas 20 y 30 imprimen un par de frases en la pantalla y sólo sirven para este ejemplo. En la línea 40 se llama a la rutina en código máquina, una y otra vez, a intervalos, con lo que se consigue el efecto FLASH. Si quieres que sea más lento o más rápido, no tienes más que aumentar o disminuir el valor 100 de los bucles FOR...NEXT. Si sólo quieres que los caracteres aparezcan en inverso, cambia la línea 40 por:

```
40 U=USR0(0):STOP
```

Si quieres ver el efecto FLASH en las minúsculas, cambia el valor 0A de la primera línea DATA, por el valor 0B. No es la rutina ideal, pero es muy corta y puede serte de gran ayuda.



LA MEMORIA DE VIDEO DE MSX: FIGURAS MOVILES

Toda la información sobre las figuras móviles o *sprites* que puede manejar tu MSX, se almacena en la memoria de vídeo (VRAM), en dos bloques o tablas de bytes. Si conoces la estructura y la función de estas dos tablas vas a poder controlar directamente los *sprites*, sin necesidad de recurrir a los comandos BASIC. Por ejemplo, con una sencilla rutina en código máquina podrás hacer que se muevan por la pantalla a velocidades de vértigo.



Los *sprites* o figuras móviles, son elementos gráficos de muchísima importancia. Aunque en realidad cada *sprite* no es más que un grupo de bytes con información sobre puntos de la pantalla, (qué puntos estarán encendidos y cuáles apagados para constituir una determinada figura) hay una ventaja fundamental en el uso de los mismos. Se trata de que quien maneja los *sprites* es nada menos que el *chip* de vídeo (el VDP), que se ocupa prácticamente de todo, actuando a gran velocidad, y quitándole al programador cantidad de trabajo. Por ejemplo, es el *chip* el que borra las imágenes anteriores de un *sprite* que se mueve, o el que da la señal de alarma cuando hay una colisión, o el que decide qué *sprite* aparece por delante cuando se superponen dos o más. Todas estas labores requerirían bastantes horas de programacion en código máquina. Además el resultado nunca sería el mismo en velocidad, al tener que ocupar tiempo de la CPU.

PARAMETROS DE LOS SPRITES

MSX permite utilizar *sprites* en los modos de pantalla 1, 2 y 3. El modo 0 es exclusivamente un modo de texto, y no admite figuras móviles.

En los modos de pantalla 1, 2 y 3 existen dos bloques de memoria VRAM que contienen la información de los *sprites*. Estos dos bloques son conocidos como **Tabla de Patrones de Figuras Moviles** y como **Tabla de Atributos de Figuras Móviles**, respectivamente.

Antes de entrar a ver qué contienen y cómo están distribuidos estos dos bloques, echemos un vistazo a los parámetros que nos interesa definir al manejar figuras móviles.

En un programa que vaya a utilizar figuras móviles hay que especificar, antes de definir éstas, el tamaño de figura móvil por medio de la sentencia SCREEN. El segundo parámetro de esta sentencia indica el tamaño de figuras móviles y puede valer 0, 1, 2 ó 3, según queramos manejar *sprites* de 8×8 puntos sin ampliar, 8×8 ampliados, 16×16 sin ampliar o 16×16 ampliados, respectivamente. Así, la sen-

tencia SCREEN 1,2 indica que en el modo 1 de pantalla se van a utilizar *sprites* de 16×16 puntos sin ampliar. Al asignar un tamaño determinado a los *sprites*, el sistema operativo actúa sobre la **Tabla de Patrones**, poniendo todos sus bytes a cero. Sin embargo,



es importante observar que en ninguna posición de la VRAM se almacena información sobre el tamaño de figuras móviles. Esta información se almacena en los registros del VDP (*Video Display Processor*).

También se almacenan en los registros del VDP la información relativa a la coincidencia de *sprites*, y a la regla del 5º *sprite* (coincidencia de 5 *sprites* en una línea horizontal).

LA TABLA DE PATRONES DE FIGURAS MOVILES

Este bloque de memoria comienza en la dirección 14336 de la VRAM, en los tres modo de pantalla que admiten *sprite*, y termina en la posición 16383, correspondiente al último byte de la VRAM. Utilizando la función BASE(n) para obtener la dirección de comienzo, vemos que n puede valer 9, 14 ó 19, según se trate del modo 1, modo 2 o del modo 3 de pantalla, por tanto:

```
BASE(9)=BASE(14)=
BASE(19)=14336
```

Vemos que este bloque tiene una longitud de 2048 bytes, y en él se almacenan los datos correspondientes a la definición, o lo que es lo mismo, a la forma de los *sprites*.

Si tenemos en cuenta que un *sprite* de 8×8 se define mediante 8 números binarios de 8 dígitos cada uno, es fácil comprobar que en estos 2048 bytes podemos almacenar los caracteres de definición de 256 figuras móviles de 8×8. Igualmente, como un *sprite* de 16×16 está formado por 4 *sprites* de 8×8, podemos almacenar, en ese espacio de memoria 64 *sprites* de 16×16 puntos.

Cómo está distribuido este bloque de memoria es algo bastante simple. Al trabajar con *sprites* de 8×8, el primero de ellos (numerado con 0) queda definido por los 8 bytes comprendidos entre el 14336 y el 14343 (ver fig. 1), el *sprite* número 1 por los 8 siguientes, del 14344 al 14351 ... y el *sprite* número 255 por los bytes 16376 al 16383. Por tanto podemos conocer la dirección de comienzo del patrón de un *sprite* cualquiera (n) con la expresión:

```
Direccion de comienzo
del sprite n=14336+8*n
```

Sin embargo (fig. 2a) al tratar con *sprites* de 16×16 puntos necesitamos 4×8=32 bytes. Por tanto, los 32 primeros bytes de este bloque de memoria (del 14336 al 14367) corresponderán al *sprite* 0, los 32 siguientes (14368 al 14399) al sprite 1,... y los 32 últimos (16352 al 16383) al *sprite* 63. La dirección de comienzo del patrón de un *sprite* (n) de 16×16 puntos viene dada por:

```
Direccion de comienzo
del sprite n=14336+32*n
```

Conviene recordar que un *sprite* de 16×16 puntos se define con 4 cuadros de 8 puntos numerados como se indica en la figura 2b. Los 32 bytes del *sprite* se reparten de la siguiente forma: los 8 primeros corresponden al cuadro número 1 (superior izquierdo), los 8 siguientes al nº 2 (inferior izquierdo), los 8 siguientes al nº 3 (superior derecho) y los 8 últimos al nº 4 (inferior derecho). La dirección de comienzo del cuadro i (i=1,2,3,4) del sprite n puede ser calculada mediante:

```
Dir. comienzo cuadro i
sprite n=14336+32*n+8*(i-1)
i=1,2,3,4
n=0,1,2,...,63
```

Después de esta descripción de la estructura de la **Tabla de Patrones**, vamos a pasar a la práctica, para ver qué es lo que podemos hacer con ella. En primer lugar vamos a presentar un ejemplo de un *sprite* intermitente. Durante un cierto intervalo aparecerá el *sprite* y en el intervalo siguiente, el negativo del mismo. Conseguiremos este negativo haciendo que, en la **Tabla de Patrones** y para el *sprite* 0, todos los bytes cambien a intervalos por sus opuestos, es decir haciendo que donde había un 1 haya un 0 y viceversa. De esto se encarga la línea 510 del programa:

```
10 SCREEN 1,0:KEY OFF
   :COLOR,1,1
20 SPRITE$(0)=CHR$(24)+CHR$
   (36)+CHR$(66)+CHR$(153)
   +CHR$(153)+CHR$(66)+CHR$
   (36)+CHR$(24)
30 PUT SPRITE 0,(110,80),
   15,0
31 PUT SPRITE 1,(110,40),
   12,0
32 PUT SPRITE 2,(110,120),
   12,0
```

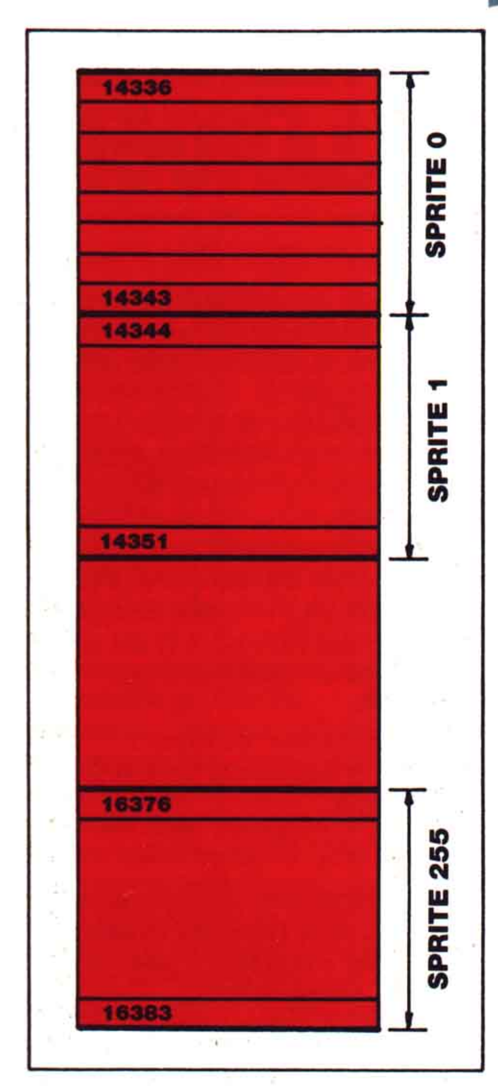


FIG. 1. TABLA DE PATRONES DE SPRITES

```
33 PUT SPRITE 3,(70,40),
   11,0
34 PUT SPRITE 4,(150,40),
   11,0
35 PUT SPRITE 5,(70,120),
   8,0
36 PUT SPRITE 6,(150,120),
   8,0
37 PUT SPRITE 7,(110,160),
   15,0
40 ON INTERVAL=20
   GOSUB 500
50 INTERVAL ON
60 GOTO 60
500 FOR I=14336 TO 14343
510 VPOKE I,255-VPEEK(I)
520 NEXT I
530 RETURN
```

Lo interesante del ejemplo es que actuamos directamente sobre la **Tabla de Patrones**, para modificar el *sprite* 0 durante la ejecución del programa. Podíamos haber hecho lo mismo definiendo dos *sprites*, uno de ellos inverso del otro, pero aparte de gastar 8 bytes más de memoria de vídeo (lo que al trabajar con muchos *sprites* puede llegar a ser un gasto importante) nos habríamos visto obligados a incluir todas las sentencias PUT SPRITE en un bucle, para cambiar de uno a otro *sprite* en cada intervalo. De este modo, trabajando directamente con la **Tabla de Patrones** de la VRAM, ahorramos espacio, hacemos el programa más corto y ganamos en velocidad.

El siguiente ejemplo es más interesante todavía. Se trata de borrar una serie de *sprites*. Teclea lo que sigue y escribe RUN.

```
10 SCREEN1,2:COLOR,1,1
   :CLS:C=12
12 FOR J=0 TO 31
14 READ A:VPOKE 14336+J,A
16 NEXT J
18 DATA 3,3,31,31,255
   ,249,255,255
20 DATA 255,220,223,223
   ,15,15,31,63
22 DATA 192,192,248,248
   ,255,159,255,255
24 DATA 255,59,251,251
   ,240,240,248,252
26 FOR I=0 TO 4
28 FOR J=0 TO 4
30 PUT SPRITE 5*I+J,
   ((90+16*I),(50+16*J)),
   C,0
32 NEXT J,I
34 FOR J=1 TO 300:NEXT
36 R=RND(-TIME)
38 FOR N=1 TO 400
40 B1=14336+INT(RND(1)*33)
   :B2=INT(RND(1)*8)
42 VPOKE B1,VPEEK(B1)AND
   (255-2^B2)
44 NEXT N
46 RESTORE:C=2+(C+1)MOD14
   :GOTO 12
```

Como verás el efecto es de lo más divertido. La rutina de borrado es la comprendida entre las líneas 34 y 44. Lo que hace es poner a cero, aleatoriamente, bits dentro del patrón del *sprite*, que ocupa los primeros 32 bytes de la tabla de patrones. Las líneas anteriores del programa se encargan únicamente de definir y dibujar en pantalla los *sprites*.

LA TABLA DE ATRIBUTOS DE FIGURAS MOVILES

Comienza en la dirección 6912 de la VRAM en los 3 modos de pantalla que admiten *sprites*, y termina en la dirección 7039. Tiene, por tanto, una longitud de 128 bytes. La dirección de comienzo se obtiene, según el modo de pantalla mediante:

```
BASE(8)=BASE(13)=
BASE(18)=6912
```

Recordemos que para situar un *sprite* en la pantalla debemos utilizar la instrucción:

```
PUT SPRITE P,(X,Y),C,N
```

en donde:

- P es el número de plano de figura móvil. (Existen 32 planos de figura móvil en **MSX**, numerados del 0 al 31, y en cada uno de ellos sólo puede haber un *sprite*).
- (X,Y) son las coordenadas del vértice superior izquierdo del *sprite*. X puede variar entre −32 y 255, e Y entre 0 y 255 (o entre −32 y 223).
- C es el color del *sprite*.
- N es el número de *sprite* a que se refiere la instrucción.

Vemos, por tanto, que se requieren 5 parámetros o ATRIBUTOS para situar un *sprite* en la pantalla.

La **Tabla de Atributos** de figuras móviles se divide en 32 trozos de 4 bytes cada uno. Cada uno de estos trozos corresponde a un plano de figura móvil. Así, entre los bytes 6912 y 6915 están los parámetros correspondientes al plano 0, entre el byte 6916 y 6919 al plano 1,... y entre el 7036 y 7039 al plano 32. La dirección de comienzo de los atributos correspondientes a un plano P viene dada por la expresión:

```
Dir. comienzo plano   (4)
P=6912+4*P
```

Si numeramos los 4 bytes correspondientes a un plano I cualquiera de 0 a 3, éstos se distribuyen así (fig. 3):

Byte 0: Ordenada (Y): Contiene el valor de Y (ordenada) de la esquina superior izquierda del *sprite*. Si se da a Y un valor que varíe de −31 a 0, el *sprite* va saliendo por la parte superior de la pantalla. En realidad este valor se almacena en complemento a 2, por lo que se obtiene el mismo efecto si de da a Y un valor comprendido entre 224 y 0 —cuando Y=256, en este byte se almacena el valor 0—). Si Y vale 208, desaparecerán todos los *sprites* que estén en planos de menor prioridad (el plano de mayor prioridad es el 0 y el de menor prioridad, el 31). Y si Y vale 209, desaparece este *sprite* de la pantalla.

Byte 1: Abcisa (X): Contiene el valor de X y puede variar entre 0 (borde izquierdo de la pantalla) y 255.

Byte 2: Número de patrón de figura móvil (N): Contiene el número que representa el patrón de figura móvil que se desea tener en el plano I. Si estamos manejando *sprites* de 8×8, los valores que se escribirán en esta posición serán 0, 1, 2,..., 255, coincidente con el número de *sprite*. Pero si los *sprites* son de 16×16 puntos, los valores que se escribirán en esta posición son 0, 4, 8,..., 252, de forma que al multiplicar este valor por 8 y sumarle 14336, se obtiene la dirección de comienzo del patrón correspondiente.

Byte 3: Los bits 0 a 3 (*nibble* inferior) de este byte contienen el color del *sprite* (C) y, lógicamente puede variar de 0 (&H0) a 15 (&HF). Los bits 4 al 6 no tienen utilidad, sin embargo, el bit 7 (bit del bloque anterior) sí. Si en este bit hay un 0, no pasa nada, pero si hay un 1, el *sprite* no se situará a partir de la posición (X,Y) sino a partir de (X−32,Y). Sirve para poder hacer qûe el *sprite* aparezca lentamente por el borde izquierdo de la pantalla (equivale a dar un valor entre −32 y 0 al parámetro X de la instrucción PUT SPRITE).

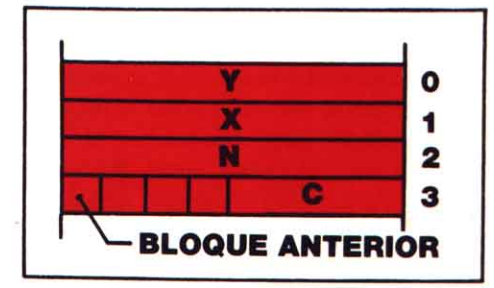


FIG. 3. ATRIBUTOS DE UN SPRITE

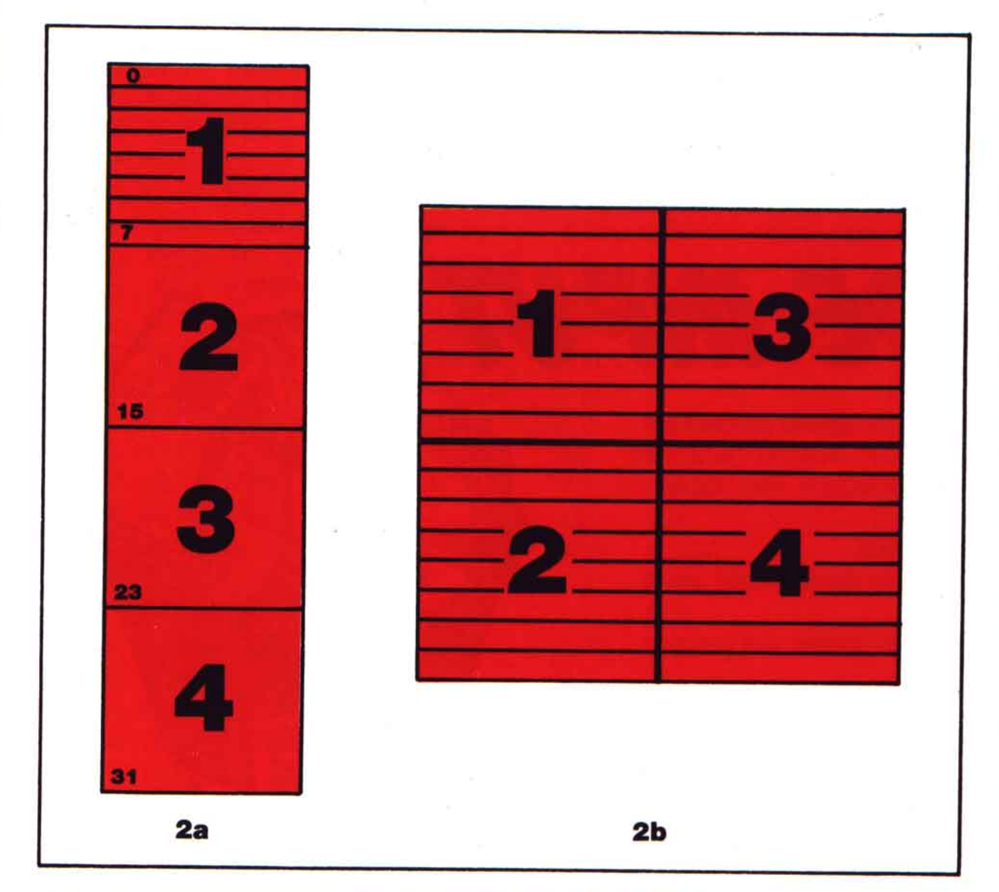


FIG. 2. DISPOSICION DE PATRONES EN SPRITES DE 16 × 16 PUNTOS

Con esto hemos visto todo lo que afecta a las figuras móviles dentro de la memoria de vídeo. Ahora ya podemos mover *sprites*, cambiarles el color o cambiar planos, sin necesidad de usar la instrucción PUT SPRITE.

Como aplicación vamos a incluir una rutina en código máquina que nos permitirá mover *sprites* a través de la pantalla usando las teclas del cursor. Esta rutina debe ser cargada a partir de la dirección 62001 de RAM. Al llamarla por medio de USR(6912), y en general, para mover el *sprite* siutado en el plano P, daremos como argumento la dirección obtenida mediante la expresión (4). Por otra parte, introduciendo un número comprendido entre 0 y 65535 en las direcciones 62300 y 62301 de RAM, podemos modificar la velocidad del movimiento del *sprite*. Para introducir este número utilizaremos la función HEX$(N) del BASIC **MSX** y almacenaremos el byte bajo en 62300 y el byte alto 62301. Si por ejemplo queremos almacenar el número 42321, haremos lo siguiente:

```
PRINT HEX$(42321).....A551
POKE 62300,&H51
POKE 62301,&HA5
```

y si queremos almacenar el número 93:

```
PRINT HEX$(93)........005D
POKE 62300,&H5D
POKE 62301,&H00
```

La rutina en C/M explora sólamente las teclas del cursor y la barra espa-

ciadora. Al pulsar ésta, la rutina devuelve el control al BASIC.

A continuación listamos la rutina en C/M (mediante sentencias DATA) y un ejemplo para comprobar la diferencia de velocidad entre BASIC y C/M. Para que el programa trabaje en BASIC basta eliminar de la línea 135 la sentencia REM, para hacer una bifurcación a la línea 5000. También hemos listado el programa ensamblador correspondiente a esta rutina. Este programa está suficientemente comentado. Sólo queda añadir que puede interesar modificar los saltos absolutos (JP y RTN) por saltos relativos (JR) para darle mayor flexibilidad) en cuanto a su ubicación en memoria.

Es preciso observar que cuando se ha dado una velocidad muy baja a la rutina en C/M (por ejemplo almacenando en 62300 y 62301 el número &H0000), ésta tarda bastante en explorar el teclado, por lo que puede parecer que no responde cuando se pulsa la barra espaciadora, no siendo así.

```
10 CLEAR 200,61500!
20 SCREEN 1,2
30 FOR I=14336 TO
   14367
40 VPOKE I,255
50 NEXT I
60 PUT SPRITE 0,(100,100)
   ,15,0
70 FOR I=1 TO 93
80 READ A$
90 POKE 62000!+I,VAL
   ("&H"+A$)
100 NEXT I
110 DEFUSR=62001!
120 POKE 62300!,&HF
130 POKE 62301!,&HO
135 'GOTO 5000
140 IF INKEY$="" THEN 140
150 U=USR(6912)
160 GOTO 140
1000 DATA F3,DB,99,DB,AA,E6,
     F0,F6,08,D3,AA,2A,5C,F3,
     2B,7C,B5,C2,3D,F2,DB,A9,
     ED,5B,F8,F7,CB,47,CA,67,
     F2,CB,7F,CA,69,F2,CB,77,
     CA,6A,F2,CB,6F,CA,72,F2,
     CB,67,CA,71,F2,C3,3C,F2,
     FB,C9
1010 DATA 13,CD,85,F2,3C,C3,
     76,F2
1020 DATA 13,CD,85,F2,3D,47,
     7B,D3,99,7A,C6,40,D3,99,
     78,D3,98,C3,3C,F2
1030 DATA 7B,D3,99,7A,D3,99,
     DB,98,C9
5000 X=100:Y=100
5005 U=STICK(0):IF U=0
     THEN 5005
5010 IF U=1 THENY=Y-1
5020 IF U=3 THEN X=X+1
5030 IF U=5 THEN Y=Y+1
5040 IF U=7 THEN X=X-1
5050 PUT SPRITE 0,(X,Y)
     ,15,0
5060 GOTO 5005
```




LISTADO ENSAMBLADOR DE LA RUTINA DE MOVIMIENTO DE SPRITES
VELOCDIDAD: PONER EN 62300 Y 62301 UN VALOR ENTRE 0 Y 65535
ENTRADA: CON USR(N) SIENDO N=DIRECCION DE VRAM DEL PARAMETRO (Y)
DE LA TABLA DE ATRIBUTOS

```
                          ORG  62001
                          LOAD 62001
F231 F3                   DI              ;DESACTIVA INTERRUPCIONES
F232 DB99                 IN   A,(99H)    ;
F234 DBAA                 IN   A,(0AAH)   ;PREPARACION LECTURA TECLADO
F236 E6F0                 AND  0F0H       ;
F238 F608                 OR   08H        ;
F23A D3AA                 OUT  (0AAH),A   ;
F23C 2A350F   LECTURA:    LD   HL,(0F35H) ;LEE VELOCIDAD
F23F 2B       RETARDO:    DEC  HL         ;RETARDO
F240 7C                   LD   A,H        ;
F241 B5                   OR   L          ;
F242 C23FF2               JP   NZ,RETARDO ;FIN RETARDO
F245 DBA9                 IN   A,(0A9H)   ;LEE TECLADO
F247 ED5BF8F7             LD   DE,(0F7F8H) ;CARGA EN DE DIR. TAB. ATRIBUT.
F24B CB47                 BIT  0,A        ;SI BARRA ESPACIOS: SALIDA
F24D CA67F2               JP   Z,SALIDA   ;
F250 CB7F                 BIT  7,A        ;SI CRSR DERECHA:   DERECHA
F252 CA69F2               JP   Z,DERECHA  ;
F255 CB77                 BIT  6,A        ;SI CRSR ABAJO:     ABAJO
F257 CA6AF2               JP   Z,ABAJO    ;
F25A CB6F                 BIT  5,A        ;SI CRSR ARRIBA:    ARRIBA
F25C CA74F2               JP   Z,ARRIBA   ;
F25F CB67                 BIT  4,A        ;SI CRSR IZQUIERDA: IZQUIERDA
F261 CA71F2               JP   Z,IZQUIERDA ;
F264 C33CF2               JP   LECTURA    ;SI NO CRSR O ESPACIO: LECTURA
F267 FB       SALIDA:     EI              ;ACTIVA INTERRUPCIONES
F268 C9                   RET             ;RETORNO AL BASIC
F269 13       DERECHA:    INC  DE         ;
F26A CD87F2   ABAJO:      CALL LEE        ;
F26D 3C                   INC  A          ;
F26E C378F2               JP   ESCRIBE    ;
F271 CD87F2   IZQUIERDA:  CALL LEE        ;
F274 CD87F2   ARRIBA:     CALL LEE        ;
F277 3D                   DEC  A          ;
F278 47       ESCRIBE:    LD   B,A        ;
F279 7B                   LD   A,E        ;
F27A D399                 OUT  (99H),A    ;PREPARACION Y SALIDA DATOS
F27C 7A                   LD   A,D        ;A VRAM
F27D C640                 ADD  A,40H      ;
F27F D399                 OUT  (99H),A    ;
F281 78                   LD   A,B        ;
F282 D398                 OUT  (98H),A    ;SALIDA DEL DATO
F284 C33CF2               JP   LECTURA    ;
F287 7B       LEE:        LD   A,E        ;
F288 D399                 OUT  (99H),A    ;PREPARACION PARA LEER VRAM
F28A 7A                   LD   A,D        ;
F28B D399                 OUT  (99H),A    ;
F28D DB98                 IN   A,(98H)    ;LEE VRAM
F28F C9                   RET             ;
                          END             ;
```

TODO LO QUE SUBE, BAJA

- TRAYECTORIAS DE UN PROYECTIL
- DIFERENTES CAMPOS GRAVITATORIOS
- SIMULACION DE PARABOLAS
- CAMBIANDO EL ANGULO

«Lancé una flecha por los aires, cayó a tierra, no sé en qué punto». Aquí tienes la forma de hacer que tu ordenador haga por tí este cálculo, además de unas cuantas rutinas que pueden servirte para construir excitantes juegos de acción.

Hay una especial belleza en la contemplación de un ordenador mientras dibuja la trayectoria de un objeto volador, y existen muchos ejemplos —especialmente en la programación de juegos— en que resulta muy adecuado disponer de programas de este tipo. Si no se pudiera disponer de un ordenador, el dibujo de este tipo de trayectorias sería extraordinariamente tedioso y plagado de errores. Sin embargo, con sólo unas cuantas líneas de programa, puedes hacer que tu micro realice con precisión la gran cantidad de cálculos necesarios para contar con resultados sorprendentes.

La forma en que se mueven los objetos puede depender de varios factores, siendo importante que en un programa de acción se tengan en cuenta al menos algunos de ellos para conseguir un aceptable efecto de realismo.

Un caso importante muy especial es el de un objeto que abandona la superficie de la Tierra y está sometido a la ley de la gravedad. En este artículo veremos la manera de programar el movimiento de los proyectiles, entendiendo como tales los objetos que se mueven con velocidad horizontal constante (despreciando el rozamiento del aire) y verticalmente sometidos únicamente a la acción de la gravedad.

Una de las razones de por qué son tan comunes los juegos de combates que se desarrollan en el espacio, no sometidos a caída libre, es que en el espacio se pueden ignorar los efectos de la gravedad y del rozamiento del aire sobre el movimiento, tanto de los vehículos espaciales como de los proyectiles. Esto no quiere decir que en el espacio libre no exista fuerza gravitacional; claro que existe, lo que ocurre es que sus efectos son normalmente tan pequeños que se pueden ignorar dentro de un margen razonable de aproximación, o que se puede suponer que todos los objetos se ven afectados por un mismo campo gravitatorio.

En contraste con lo anterior, las batallas sobre la Tierra que implican el lanzamiento de proyectiles, han de tener en cuenta la gravedad y con frecuencia también la resistencia del aire y la acción del viento. Aparte de los juegos de tipo militar, hay otros muchos ejemplos en los que un conocimiento del movimiento de los proyectiles resulta esencial para una programación realista. Entre dichos ejemplos se encuentran muchos deportes y simulaciones de tipo atlético, incluyendo el lanzamiento de pelotas, el tiro con arco y todo tipo de saltos.

Todos los proyectiles de estos ejemplos (ya se trate de personas o cosas) tienen una cosa en común: la trayectoria descrita responde a una familia de curvas conocidas como parábolas. No es difícil escribir un programa que simule el movimiento según una trayectoria parabólica; para ello no se requiere más que un entendimiento de las fuerzas que originan el movimiento y el uso de un poco de matemáticas elementales.

Por ejemplo, basta con saber que el movimiento de un balón al que se le da un puntapié, consiste en la combinación de dos movimientos en dos direcciones distintas, para poder abordar y resolver el problema de la programación de este movimiento. Una de estas direcciones es la del eje horizontal o eje X. La otra dirección es la del eje vertical o eje Y. A través de todo este artículo se supone que un proyectil se mueve a velocidad constante en la dirección del eje X. En realidad un objeto real se vería frenado por la acción de la resistencia del aire, pero esto es una complicación que es mejor evitar en general salvo en los trabajos de gran precisión.

MOVIMIENTO HORIZONTAL

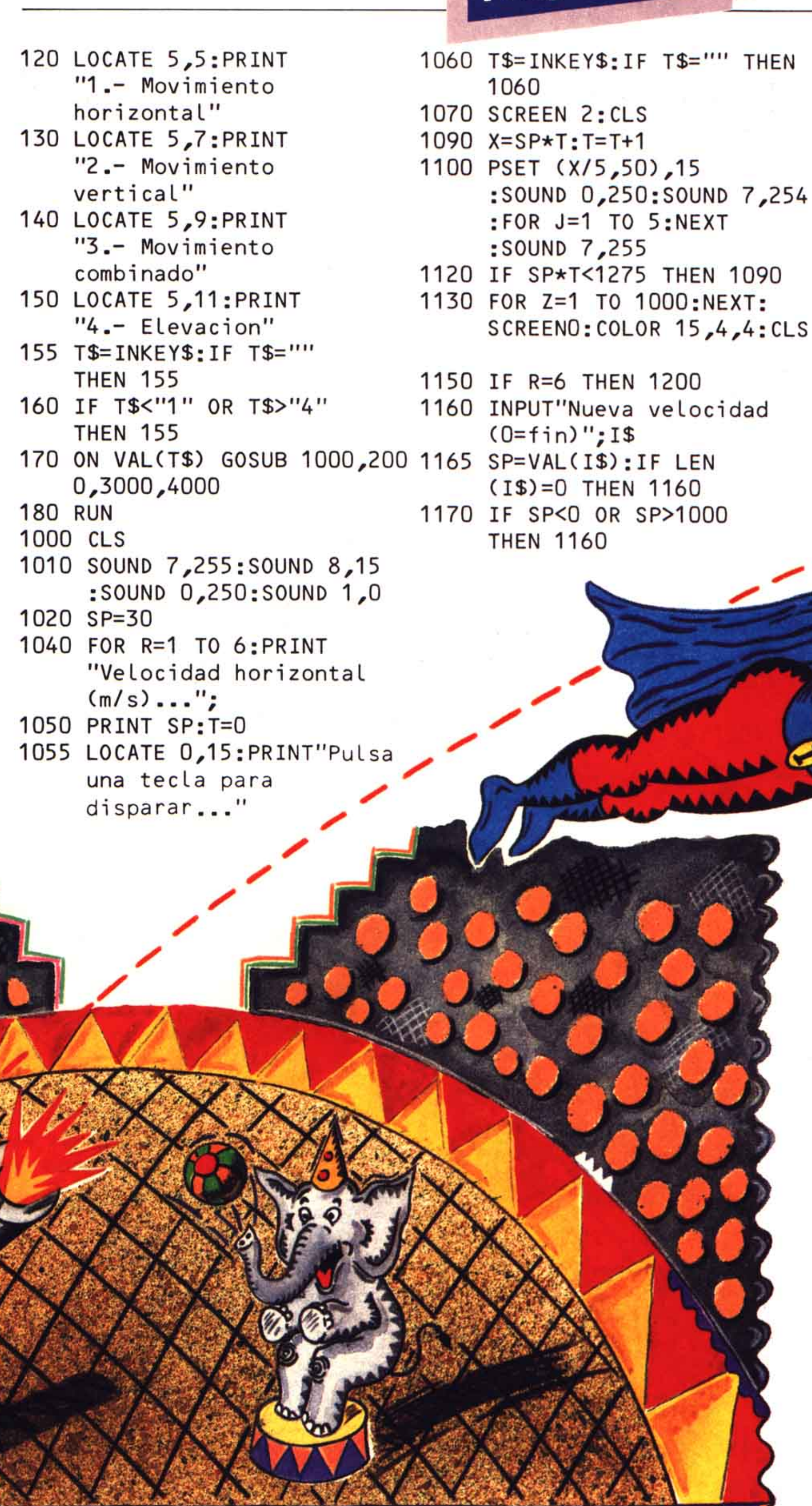
Teclea el siguiente programa para simular el movimiento horizontal, pero no confundas «I» con «1».

```
100 SCREEN0:KEY OFF:COLOR
    15,4,4:CLS
110 LOCATE 8,0:PRINT"MENU DE
    OPCIONES":LOCATE 8,1
    :PRINT"---- -- --------"
```



```
120 LOCATE 5,5:PRINT
    "1.- Movimiento
    horizontal"
130 LOCATE 5,7:PRINT
    "2.- Movimiento
    vertical"
140 LOCATE 5,9:PRINT
    "3.- Movimiento
    combinado"
150 LOCATE 5,11:PRINT
    "4.- Elevacion"
155 T$=INKEY$:IF T$=""
    THEN 155
160 IF T$<"1" OR T$>"4"
    THEN 155
170 ON VAL(T$) GOSUB 1000,200
    0,3000,4000
180 RUN
1000 CLS
1010 SOUND 7,255:SOUND 8,15
     :SOUND 0,250:SOUND 1,0
1020 SP=30
1040 FOR R=1 TO 6:PRINT
     "Velocidad horizontal
     (m/s)...";
1050 PRINT SP:T=0
1055 LOCATE 0,15:PRINT"Pulsa
     una tecla para
     disparar..."
1060 T$=INKEY$:IF T$="" THEN
     1060
1070 SCREEN 2:CLS
1090 X=SP*T:T=T+1
1100 PSET (X/5,50),15
     :SOUND 0,250:SOUND 7,254
     :FOR J=1 TO 5:NEXT
     :SOUND 7,255
1120 IF SP*T<1275 THEN 1090
1130 FOR Z=1 TO 1000:NEXT:
     SCREEN0:COLOR 15,4,4:CLS

1150 IF R=6 THEN 1200
1160 INPUT"Nueva velocidad
     (0=fin)";I$
1165 SP=VAL(I$):IF LEN
     (I$)=0 THEN 1160
1170 IF SP<0 OR SP>1000
     THEN 1160
1190 IF SP=0 THEN RETURN
1200 NEXT R
1210 RETURN
```



Al ejecutar el programa te encuentras con un menú de cuatro opciones. De momento sólo tienes tecleada la rutina correspondiente a la primera opción, por lo que si pulsas 2, 3 o 4 te aparecerá un mensaje de error. Al teclear 1, el programa salta a la rutina que comienza en la línea 1000.

Dicha rutina utiliza un bucle FOR ... NEXT (líneas 1040 a 1200) que te permite lanzar un objeto horizontalmente a seis velocidades distintas. La primera vez que se recorre el bucle, se elige automáticamente una velocidad de 30 metros por segundo (m/s), la cual se simula como una serie de puntos a lo largo de una línea recta.

Después de que se ha hecho la primera pasada del bucle, el programa te invita a teclear un nuevo valor para la velocidad, pero puedes salir del bucle tecleando un 0 cuando el programa se ejecute de nuevo y presente el menú. Teclea un valor, por ejemplo 60, y pulsa RETURN de nuevo para disparar. Compara ahora el resultado con la línea de los 30 m/s. Introduce otros cinco valores de la velocidad, tras de lo cual la pantalla volverá a presentar el menú.

La sección de programa que se ocupa de dibujar los puntos, se extiende entre las líneas 1090 y 1120. La variable T simula el tiempo, que en este caso aumenta en pasos de un segundo, de forma que los puntos quedan regularmente espaciados en la dirección horizontal (eje X), lo cual es ne-

cesario por ser la velocidad constante. La separación real de los puntos se define por medio del término SP*T de la línea 1090. Este término asegura que cuanto mayor es la velocidad (SP), mayor es la separación entre puntos.

Puede que ya te hayas dado cuenta de que SP*T forma parte de la conocida fórmula Espacio = Velocidad × Tiempo, que se estudia en física elemental. A partir de aquí, resulta claro que el programa dibuja distancias (el espaciado entre puntos) para simular la velocidad.

MOVIMIENTO VERTICAL

Teclea ahora las siguientes líneas, que te proporcionarán una rutina para simular el movimiento en la dirección vertical:

```
2000 CLS
2010 SOUND 7,255:SOUND 8,15
2020 G=10:SP=50
2040 FOR R=1 TO 6:SCREEN 0
2045 PRINT"Aceleracion
     gravedad (m/s/s)...";
2050 PRINT G
2055 LOCATE 0,20:PRINT
     "Pulsa una tecla..."
2060 T$=INKEY$:IF T$=""
     THEN 2060 ELSE SCREEN2
     :CLS
2080 FOR T=0 TO 250 STEP .5
2090 H=SP*T-.5*G*T*T
2100 IF H>1000 THEN
     GOTO 2104
2102 IF H>=0 THEN PSET
     (100,190-H/5)
2104 IF H>=0 AND (H/10)<255
     THEN SOUND 0,255-H/10
     :SOUND 7,254:FOR J=1
     TO 5:NEXT:SOUND 7,255
     :NEXT T
2110 IF R=6 THEN GOTO 2180
2140 FOR K=1 TO 1000:NEXT
     :SCREEN0:CLS:INPUT
     "Nueva g (0=fin)";I$
2142 IF LEN(I$)=0 THEN 2140
2145 G=VAL(I$)
2150 IF LEN(I$)=0 OR G<0
     OR G>400 THEN 2140
2170 IF G=0 THEN R=6
2180 NEXT R
2190 RETURN
```

Haz correr el programa, tecleando ahora un 2 para seleccionar la segunda opción del menú. Pulsa RETURN y verás una serie de puntos dibujados verticalmente sobre la pantalla. Date cuenta no obstante de que los puntos no están regularmente espaciados, sino que se encuentran más agrupados cerca de la parte alta.

Esto se debe a que el movimiento del objeto se encuentra retardado por la acción de la gravedad. En esta ocasión el sonido te ayuda a explicar lo

que está sucediendo. A medida que el objeto asciende, el tono del sonido se va elevando, mientras que cuando cae, su tono se va haciendo más bajo.

Igual que antes, la rutina te permite seis pruebas (línea 2040) con introducción de diferentes valores para apreciar el efecto de la gravedad. Esta es la misión de la variable G, cuyo valor se pone inicialmente a 10 (línea 2020) y se utiliza en la línea 2090.

Esta relación es la contenida en la fórmula de un objeto que cae sometido a la acción de la gravedad. Su forma usual es $s = 1/2gt^2$, donde s es el espacio recorrido (en este caso H), t el tiempo y g el valor de la aceleración de la gravedad (G). Observa que en la línea 2090 se ha escrito T*T en lugar de T^2, ya que los ordenadores hacen con más rapidez la multiplicación que la elevación a potencias.

La aceleración de un objeto es una medida de lo que cambia su velocidad con el paso del tiempo. Cerca de la superficie de la Tierra, g tiene un valor aproximadamente igual a 10 m/s/s. Esto significa que la velocidad de un cuerpo que cae aumenta a razón de 10 m/s en cada segundo. Se trata de una aceleración negativa, ya que la línea 2090 contiene un signo menos (en lugar del signo más que aparece en la fórmula usual).

La expresión de g suele utilizarse habitualmente en relación con los viajes espaciales. Por ejemplo, la aceleración de un vehículo espacial tripulado que se lanza para entrar en órbita es próxima a los 10 g. Esto significa que dicho vehículo espacial aumenta su velocidad a razón de 10*10 m/s/s, es decir 100 m/s^2. En un posterior artículo nos ocuparemos con mayor detalle del tema de las órbitas.

La primera vez que se recorre el bucle que empieza en la línea 2040, se dibujan las posiciones del objeto a intervalos de un segundo. La velocidad inicial es de 50 metr s por segundo y el valor de G es de 10 m/s^2. Puedes cambiar entonces el valor de G, cuando te lo indique el mensaje de pantalla, y comparar el efecto de seis valores diferentes. Antes de que se complete el bucle, puedes salirte de él introduciendo un 0 cuando el programa retorna al menú.

MOVIMIENTOS COMBINADOS

Para simular el movimiento de un proyectil, sólo tienes que combinar estas rutinas de forma que el objeto se mueva al mismo tiempo en las dos direcciones, horizontal y vertical. Teclea las siguientes líneas, y tendrás la tercera rutina:

```
3000 CLS
3010 SOUND 7,255:SOUND 8,15
3020 G=10:SP=80
3030 FOR R=1 TO 6:SCREEN 0
3040 PRINT"G=";G;"(m/s/s)"
     :PRINT"VEL=";SP;"(M/S)"
3045 LOCATE 0,20:PRINT
     "Pulsa una tecla..."
3050 T$=INKEY$:IF T$=""
     THEN 3050 ELSE SCREEN 2
3070 FOR T=0 TO 100 STEP .5
3080 H=SP*SIN((3.14159/180)
     *45)*T-.5*G*T*T
3090 X=SP*COS((3.1415/180)
     *45)*T
3100 IF H>545 AND X>1200
     THEN 3105
3102 IF H>0 THEN PSET
     (X/5,190-H/5),15
3105 IF H/5>254 THEN T=100
     :GOTO 3110
3106 IF H>=0 THEN SOUND
     0,255-H/5:SOUND 7,254
     :FOR D=1 TO 2:NEXT D
     :SOUND 7,255
3107 IF H<0 THEN T=100
3110 NEXT T
3115 IF R=6 THEN GOTO 3230
3120 FOR J=1 TO 1000
     :NEXT J:SCREEN0
3140 INPUT"Nueva g (0=fin)"
     ;I$
3150 IF LEN(I$)=0 THEN 3140
3155 G=VAL(I$)
3160 IF LEN(I$)=0 OR G<0 OR
     G>1000 THEN 3140
3170 IF G=0 THEN R=6:GOTO
     3230
3190 INPUT"Nueva vel.";I$
3195 SP=VAL(I$)
3200 IF LEN(I$)=0 OR SP<1 OR
     SP>1000 THEN 3190
3230 NEXT R
3240 RETURN
```

Ejecuta el programa, tecleando un 3 como respuesta al mensaje de pantalla, seleccionando así la tercera opción. Cuando pulses RETURN, van apareciendo puntos en una curva que arranca en la parte inferior izquierda de la pantalla y termina en algún punto de la parte inferior derecha. Esta es la trayectoria de un objeto disparado a una velocidad de 80 m/s con un valor de la gravedad de 10 m/s^2.

La estructura de la rutina es análoga a las de las dos anteriores. Las secciones encargadas del cálculo y del dibujo están en un bucle FOR ... NEXT (líneas 3030 a 3230), que te permite hacer una comparación de seis trayectorias posibles, cinco de las cuales son especificadas por tí. Al igual que en las pruebas anteriores, puedes salirte de la rutina tecleando un 0 para volver al menú. Sin embargo es más probable que quieras introducir un nuevo juego de valores para comparar las trayectorias.

Teclea un valor de 5 para G, manteniendo el valor de SP en 80, y a continuación pulsa RETURN para disparar. Esta vez el objeto se moverá con mayor rapidez y llegará más lejos. Mantén ahora el valor de G a 5, pero reduciendo SP a 25 y compara los resultados. Continúa la experimentación cambiando G y SP, escuchando los sonidos, para ayudarte a entender el movimiento del objeto cuando desaparece de la pantalla. Una nota cuya altura aumenta indica que el objeto está subiendo, mientras que cuando el tono baja, indica que el objeto está cayendo.

FUNCIONAMIENTO

Recuerda que la trayectoria del objeto se dibuja con coordenadas H en la dirección del eje Y, y coordenadas X en la dirección del eje X. Estas dos coordenadas se calculan en las líneas 3080 y 3090. La única diferencia es que aquí la coordenada H contiene la velocidad (SP) multiplicada por el seno de un ángulo, y la coordenada X contiene la SP multiplicada por el coseno de ese mismo ángulo (45°). Esto explica porqué cuando G es pequeño y SP es grande, la trayectoria aparente es una línea diagonal a 45°.

La razón por la que se emplean estas funciones trigonométricas es el cálculo de la fracción del movimiento del objeto que corresponde a cada una de las dos direcciones: vertical y horizontal. A estas fracciones se les llama las componentes. Si por ejemplo la velocidad inicial del objeto es 50 m/s, las componentes horizontal y vertical son menores que 50, si bien su suma da exactamente 50 m/s.

No te equivocarás si recuerdas que la componente vertical es SP*SIN A y la componente horizontal es SP*COS A. La variable A es el ángulo de elevación del cañón, el arco o lo que sea.

Un proyectil inicia su movimiento con una velocidad real V y un ángulo A con la horizontal.

El seno del ángulo A es Vh/V, relación que puede escribirse de forma equivalente como Vh = V*SIN A. Análogamente, el coseno de A es Vx/V, lo que es equivalente a escribir Vx = V*COS A.

Siguiendo con esta misma idea en nuestro programa, necesitas calcular SP*SIN 45 para la componente vertical de la velocidad y SP*COS 45 para la componente horizontal.

CAMBIO DE ANGULO

Llegados a este punto, puede que te extrañe el por qué ha de fijarse el valor del ángulo en 45°, ya que ello limita el alcance del proyectil. Tanto el ángulo de elevación como la velocidad inicial pueden modificarse para cambiar el alcance de los proyectiles. Y es más corriente modificar únicamente el ángulo para cambiar la distancia, manteniendo constante la velocidad inicial. Esto es precisamente lo que hace la siguiente rutina:

```
4000 CLS
4010 FL=0:SOUND 7,255
     :SOUND 8,15
4040 A=70:SP=50
4060 SCREEN0:PRINT
     "Elevacion=";A
4062 LOCATE0,20:PRINT
     "Pulsa una tecla"
4065 T$=INKEY$:IF T$=""
     THEN 4065
4070 SCREEN2
4080 FOR T=0 TO 250 STEP .5
4090 H=SP*SIN(3.1415/180*A)*
     T-.5*10*T*T:X=50*COS
     (3.1415/180*A)*T
4100 IF H<400 AND X<600
     THEN PSET
     (30+X/1.5,190-H/1.5)
4102 IF H/5<255 AND H>0 THEN
     SOUND 0,255-H/5
     :SOUND 7,254:FOR J=1
     TO 2:NEXT:SOUND 7,255
4105 IF H<0 OR H/5>255
     THEN T=250
4107 NEXT T
4110 FOR Z=0 TO 1000:NEXT Z
```



```
      :SCREEN0
4130 INPUT"Nueva elevacion
      (0=fin)";I$
4135 IF LEN(I$)=0 THEN 4130
4140 A=VAL(I$)
4150 IF LEN(I$)=0 OR A<0 OR
      A>90 THEN 4130
4160 IF A=0 THEN RETURN
4170 GOTO 4060
```

Al ejecutar la cuarta opción, verás la trayectoria seguida por un objeto lanzado con una velocidad de 50 m/s y con un ángulo de elevación de 70° (ambos valores se establecen en la línea 4040).

Esta rutina funciona igual que la anterior, si bien aquí puedes introducir los valores de los ángulos que quieras sin tener que volver al menú. El ángulo ha de estar comprendido entre 1° y 89°. Así, cada vez que ejecutes la rutina, se le asigna un valor a la variable A de la línea 4090, correspondiente al ángulo que hayas tecleado. Esta vez la rutina contiene un bucle infinito, por lo que la pantalla no retornará al menú a menos que introduzcas 0 como valor del ángulo.

Utilizando esta rutina, trata de encontrar el ángulo con el que se obtiene el mayor alcance, es decir el que permite que el objeto llegue más lejos en dirección horizontal. Fácilmente descubrirás que este ángulo es de 45 grados.

Casi todo el mundo conoce este resultado de forma intuitiva o a partir de su propia experiencia, pero lo que ya es menos conocido es que en la práctica la resistencia del aire hace que haya una diferencia significativa en el resultado en muchos casos, y que el valor de 45 grados sólo permite obtener el alcance máximo cuando los puntos de partida y de caída están a la misma altura. Sin embargo, cualquier juego de proyectiles que dependa sólo de que el jugador obtenga el mayor alcance, está condenado al fracaso, ya que no se producirá una variación sustancial respecto al caso ideal que la mayoría de los jugadores conocen o sospechan. En un artículo posterior veremos la forma en que pueden utilizarse estas rutinas como base para un desafiante y cautivador juego.



ASOMATE A LAS VENTANAS

Las ventanas son un interesante recurso del programador para conseguir una buena presentación en pantalla de tablas, menús o cualquier otro tipo de información.

Cada día es mayor el número de programas que las incorporan. Vamos a experimentar un poco con ellas, a través de unas cuantas rutinas en código máquina, que podrás incorporar a tu biblioteca de rutinas interesantes.

Las ventanas, lo mismo que los menús de iconos o el uso de un ratón, son elementos que se incorporan a los programas e incluso a los sistemas operativos de las modernas máquinas, con el objetivo de hacerlos más sencillos de manejar, más asequibles y atractivos para cualquier usuario.

En los paises de habla inglesa, de donde suelen llegar las novedades en programación, se utiliza el término «*User Friendly*» que signfica algo así como «Amigo del Usuario» para definir a todos estos elementos, que tratan de hacer más agradable cualquier programa, haciéndolo atractivo para más gente y consiguiendo que los fabricantes del programa vendan más y más copias del mismo.

En este artículo vamos a explorar el mundo de las ventanas, intentando comprender cómo funcionan y preparando algunas rutinas para **MSX** que nos permitan incluir ventanas en cualquier programa.

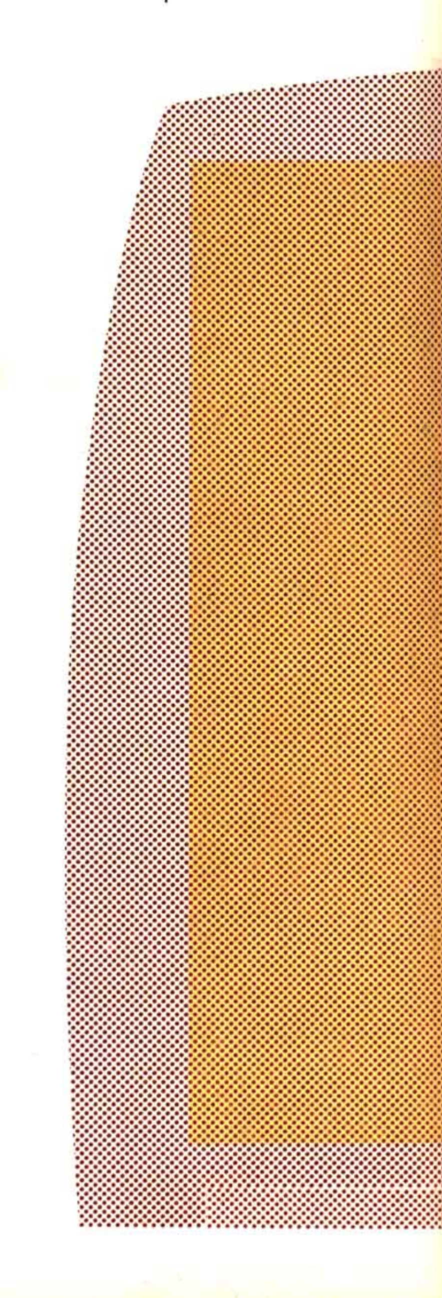
¿QUE ES UNA VENTANA?

No es fácil definir con exactitud lo que es una ventana, pero podemos acercarnos al concepto pensando en ella como en una imagen, normalmente un rectángulo, que se superpone a la imagen normal de la pantalla. A través del rectángulo de la ventana el programa puede presentar cualquier cosa, desde un menú hasta una lista de datos pasando por cualquier tipo de gráficos, texto, etc.

Sin embargo, el elemento esencial de toda ventana es la mencionada superposición. Al abrir o activar una ventana, aparece una imagen en la pantalla superponiéndose y borrando, en parte o por completo, la imagen que había anteriormente. Pero esa parte de imagen borrada (tapada por la ventana), no se pierde, queda almacenada hasta el momento en que se cierra o desactiva la ventana. Entonces vuelve a aparecer completando y dejando intacta la imagen original, como si nunca se hubiera modificado. Esta es la característica fundamental que más claramente define a una ventana: al cerrarse la misma volvemos a encontrarnos con la pantalla que había anteriormente.

Dentro de esta definición más o menos general, podemos incluir el hecho de que la ventana se tiene que poder abrir y cerrar en cualquier momento, independientemente de lo que haya en pantalla. Se podría pensar en un programa en el que, en un determinado momento, pero sólo en ese momento, fuera posible abrir la ventana. En este caso, más que de una ventana, se trataría de un simple formateo de pantalla, más o menos elaborado. En lo que vamos a ver la ventana se podrá abrir y cerrar en cualquier momento, sin importar el tipo de imagen que aparezca en la pantalla, eso sí, teniendo en cuenta que sólo vamos a trabajar en el modo de pantalla 0 que es un modo de texto, sin gráficos. Además de poder abrir y cerrar la ventana en cualquier momento, te enseñaremos cómo incluir texto en ella, cómo elegir y modificar sus dimensiones y el lugar que ocupa en la pantalla.

Dentro del concepto de ventana hay que considerar algunas variaciones. En algunos programas, los más sofisticados, la ventana puede actuar como una segunda pantalla, independiente de la pantalla principal, sobre la que aparecen textos o imágenes con su *scrolling*, su formteo, etc. Pero con la particularidad de que mientras está abierta la ventana, la pantalla principal sigue funcionando con normalidad, presentando textos o cualquier otro dato, en todas partes menos en el



interior de la ventana. Con este esquema tan sofisticado, podemos encontrarnos con programas en los que hay varias ventanas abiertas, además de la pantalla principal, y en todas ellas están presentándose datos correspondientes a distintas fases de un programa o incluso a programas distintos. Este tipo de ventanas quedan fuera de nuestras pretensiones, al menos por el momento. Las que te vamos a ofrecer en este artículo, aunque permiten, mientras están abiertas, que el programa siga funcionando, no admiten que se modifique la pantalla principal. Para modificarla será necesario que antes cierres la ventana. De todas formas, estas rutinas pueden ser un punto de partida para que algún que otro manitas se diseñe un sistema complejo con multiples ventanas funcionando a la vez.

ABRIENDO Y CERRANDO VENTANAS

El primer paso que vamos a dar es el de abrir la ventana. Pensemos un poco en lo que tenemos que hacer. Abrir la ventana significa escoger una zona de la pantalla y borrarla, pero teniendo buen cuidado de almacenar en algún sitio lo que vamos a borrar, para, a la hora de cerrar la ventana, poder recuperarlo recomponiendo la pantalla original. Para ello tenemos que trabajar con la memoria de vídeo (la VRAM) que es una zona de memoria en donde se almacena lo que aparece en la pantalla. Vamos a trabajar en el modo 0 de pantalla, el modo normal cuando no se utilizan gráficos. En este modo, la pantalla se encuentra dividida en una serie de casillas, en cada una de las cuales puede aparecer un caracter. Las casillas se organizan en 24 filas de 40 columnas, lo que nos da un total de 960 casillas, que podemos numerar de 0 a 959 empezando por la de la esquina superior izquierda y siguiendo casilla a casilla hacia la derecha y hacia abajo. En la VRAM y en este modo de pantalla,



casilla cero se corresponde con el byte cero, la uno con el byte uno y la 959 con el byte 959. En cada uno de estos bytes se almacena el código ASCII del caracter que aparece en la pantalla, en la casilla correspondiente al byte. Por ejemplo, si en uno de los bytes de VRAM almacenamos el valor 65, en la casilla correspondiente de la pantalla aparecerá el caracter A, que corresponde al código ASCII 65. Si lo que almacenamos es el valor 32, aparecerá en la pantalla un espacio (el código ASCII del espacio es 32). Y lo mismo ocurre a la inversa, si leemos un byte de VRAM correspondiente a una casilla en la que aprezca una A, el valor leído será 65. Si quieres prueba a escribir desde BASIC:

VPOKE valor 1,valor2

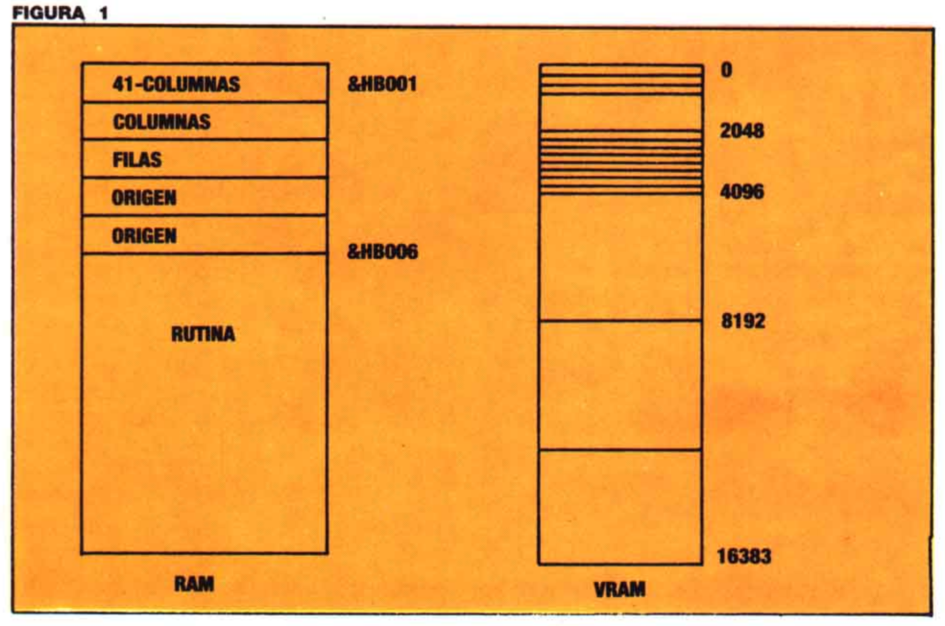
siendo valor 1 un número de 0 a 959 y valor 2 otro de 0 a 255. Verás como aparece el caracter correspondiente en la pantalla.

Con esta organización, para abrir nuestra ventana no tenemos más que elegir donde la vamos a colocar, calcular que bytes de VRAM corresponden a la ventana, copiar estos bytes en otra zona de memoria (para poder recuperarlos al cerrar la ventana) y una vez copiados y a salvo, poner en todos ellos el valor 32 (espacio). De esta forma aparecerá una zona limpia en la pantalla: la ventana.

```
                              ;*****ABRIR*****
                              ;
                                           ORG   0B006H
                                           LOAD  0B006H
                              RVRAM:       EQU   4AH
                              WVRAM:       EQU   4DH
                              ;
B006 CD5DB0                                CALL  LEPA
                              ;
B009 CD4A00                   BU1:         CALL  RVRAM
B00C E5                                    PUSH  HL
B00D F5                                    PUSH  AF
B00E 7C                                    LD    A,H
B00F F620                                  OR    20H
B011 67                                    LD    H,A
B012 F1                                    POP   AF
B013 CD4D00                                CALL  WVRAM
B016 E1                                    POP   HL
B017 3E20                                  LD    A,20H
B019 CD4D00                                CALL  WVRAM
B01C 1D                                    DEC   E
B01D C228B0                                JP    NZ,BU2
B020 FD5E01                                LD    E,(IY+1)
B023 15                                    DEC   D
B024 C22CB0                                JP    NZ,BU3
B027 C9                                    RET
B028 23                BU2:                INC   HL
B029 C309B0                                JP    BU1
B02C FD4E00            BU3:                LD    C,(IY+0)
B02F 0600                                  LD    B,00H
B031 09                                    ADD   HL,BC
B032 C309B0                                JP    BU1
```

Todas estas operaciones consumen tiempo. Realizarlas en BASIC, aunque posible, resultaría muy lento y como lo importante es que la ventana se abra a toda velocidad, no nos queda más remedio que recurrir al código máquina.

FIGURA 1



La primera rutina, que abrirá y cerrará una ventana, vamos a colocarla en memoria (mediante un cargador BASIC) a partir de la dirección &HB006. Las direcciones &HB001 a &HB005, las utilizaremos para pasarle a la rutina como parámetros, la dirección de la esquina superior izquierda de la ventana, el número de líneas y el número de columnas (ver la figura 1). Utilizando estas direcciones para el paso de parámetros, podemos hacer que se comuniquen la rutina y cualquier programa BASIC. Se trata

```
                        ;*****CERRAR*****
                        ;
B035 CD5DB0                     CALL LEPA
                        ;
B038 E5                 L1:     PUSH HL
B039 7C                         LD   A,H
B03A F620                       OR   20H
B03C 67                         LD   H,A
B03D CD4A00                     CALL RVRAM
B040 E1                         POP  HL
B041 CD4D00                     CALL WVRAM
B044 1D                         DEC  E
B045 C250B0                     JP   NZ,L2
B048 FD5E01                     LD   E,(IY+1)
B04B 15                         DEC  D
B04C C254B0                     JP   NZ,L3
B04F C9                         RET
B050 23                 L2:     INC  HL
B051 C338B0                     JP   L1
B054 FD4E00             L3:     LD   C,(IY+0)
B057 0600                       LD   B,00H
B059 09                         ADD  HL,BC
B05A C338B0                     JP   L1
                        ;
                        ;*****LECTURA PARAMETROS*****
                        ;
B05D FD2101B0           LEPA:   LD   IY,0B001H
B061 FD5E01                     LD   E,(IY+1)
B064 FD5602                     LD   D,(IY+2)
B067 FD6E03                     LD   L,(IY+3)
B06A FD6604                     LD   H,(IY+4)
B06D C9                         RET
                                END
```

de una práctica habitual al trabajar en código máquina. El programa BASIC, mediante instrucciones POKE, deja determinados valores en determinadas direcciones de memoria. La rutina en código máquina acude a dichas direcciones y puede leer los valores que dejó el programa BASIC. La cosa funciona igual de bien a la inversa. La rutina para abrir ventanas, primero leerá los valores que le pasa el programa BASIC y a continuación se dedicará a leer los valores correspondientes de la VRAM, los almacenará en alguna zona de memoria e introducirá, en lugar del valor leído, el valor 32, correspondiente al espacio. La pregunta que surge es: dónde colocar lo que hemos leído en la VRAM. Una solución sería copiarlo en memoria RAM, en una zona reservada de 960 bytes (para que cupiera toda la pantalla), pero hay una solución más interesante. Si nos fijamos en la figura 1. veremos que de la VRAM, que tiene nada menos que 16384 bytes , sólo se utilizan, en el modo 0 de pantalla, los 960 primeros bytes, como hemos visto cada uno para una casilla de la pantalla, y otros 2048 bytes, (entre el 2048 y el 4096) para la tabla generadora de caracteres, donde se guarda la información de cómo es cada caracter. Así que, a partir de la dirección 4096 y hasta la 16383, tenemos una buena zona de VRAM libre. Es ahí donde vamos a guardar la copia de la pantalla. De este modo la rutina en código máquina no tiene más que leer bytes de VRAM y copiarlos en otra zona, también de VRAM. De esta forma no desperdiciamos memoria RAM que nos vendrá bien para los programas BASIC.

Hemos dicho que, además de abrir la ventana, la rutina nos va a permitir cerrarla. Con lo que hemos contado hasta ahora, comprenderás enseguida en qué consiste cerrar la ventana. Se trata simplemente de realizar el proceso inverso al de abrirla. Esto es algo tan simple como copiar un bloque de datos de una zona a otra de la VRAM. En efecto, al abrir la ventana hemos almacenado lo que ésta iba a borrar, en unos cuantos bytes de VRAM a partir de la dirección 4096. Entonces, para cerrar la ventana, no tenemos más que devolver el contenido de estos bytes a su posición en la pantalla. No hay ningún problema en cuanto a la posición que van a ocupar ya que se recuperan exactamente en el mismo orden (mediante el mismo tipo de bucle) en que se guardaron. La rutina de cerrar es algo más simple que la de abrir. La razón estriba en que al abrir, teníamos que guardar la zona de pantalla que iba a ser ocupada por la ventana. Ahora al cerrar no tenemos esa preocupación, pues lo que vamos a borrar es la propia ventana. Esta no hace falta que la almacenemos en ningún sitio. Es siempre la misma ventana y basta con que la almacenemos una vez, al comienzo del programa (esto es lo que hacemos al definir el origen, el número de filas y el número de columnas).

Visto dónde colocamos la rutinas, dónde vamos a colocar la copia de pantalla y cómo vamos a pasar la información de las dimensiones de la ventana desde BASIC, no tenemos más que escribir la rutina y aprender a utilizarla. En primer lugar te ofrecemos el diagrama de flujo de la rutina (figura 2). A partir de este diagrama obtuvimos el listado en ensamblador que también te ofrecemos (en el que hemos marcado las tres rutinas principales de ABRIR, CERRAR y LEER PARAMETROS) y a partir de él llegamos al cargador BASIC que es éste que sigue:

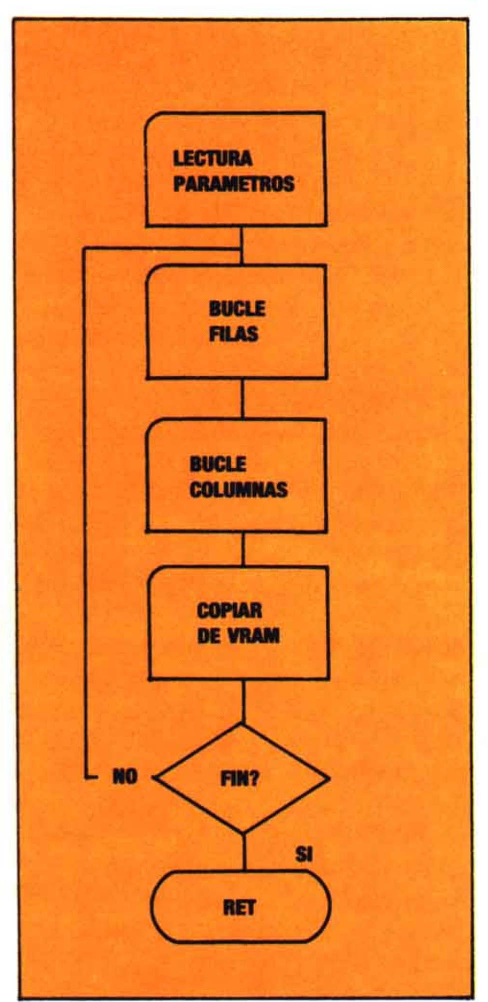


FIGURA 2

```
10 CLEAR 200,&HB000:GOSUB
   1000:DEF USR0=&HB006:
   DEF USR1=&HB035
12 '
14 'ORIGEN DE LA VENTANA
16 POKE &HB004,251
18 POKE &HB005,0
20 '
22 'NUMERO DE FILAS
24 POKE &HB003,10
26 '
28 'NUMERO DE COLUMNAS
30 POKE &HB002,12
32 '
34 'DIFERENCIA=41-NUMERO
   DE COLUMNAS
36 POKE &HB001,29
38 '
100 COLOR15,4,4:WIDTH40:CLS
102 LOCATE 8,0:PRINT
    "ASOMATE A LAS VENTANAS"
104 PRINT:PRINT"Lo que sigue
    podria ser el texto de
    pantalla de cualquier
    programa BASIC":PRINT
106 FOR J=1 TO 17:PRINT 5728+
    J;:PRINT"ESTA PUEDE SER L
    A FRASE";J
108 NEXT
120 T$=INKEY$:IF T$=""
    THEN 120
130 IF T$="A" THEN U=USR0(0)
140 IF T$="F" THEN GOTO 100
142 IF T$="C" THEN U=
    USR1(0)
150 GOTO 120
1000 FOR N=&HB006 TO &HB06D
1002 READ V$:V=VAL("&H"+V$)
1004 POKE N,V
1006 NEXT
1008 RETURN
2000 DATA CD,5D,B0,CD,4A,00,
     E5,F5,7C,F6,20,67,F1,CD,
     4D,00,E1,3E,20,CD,4D,00,
     1D,C2,28,B0,FD,5E,01,15,
     C2,2C,B0
2002 DATA C9,23,C3,09,B0,FD,
     4E,00,06,00,09,C3,09,B0,
     CD,5D,B0,E5,7C,F6,20,67,
     CD,4A,00,E1,CD,4D,00,1D,
     C2,50,B0
2004 DATA FD,5E,01,15,C2,54,
     B0,C9,23,C3,38,B0,FD,4E,
     00,06,00,09,C3,38,B0,FD,
     21,01,B0,FD,5E,01,FD,56,
     02,FD,6E,03
2006 DATA FD,66,04,C9
```

Tecléalo, asegúrate que está correcto y guárdalo en *cassette* o en disco antes de escribir RUN. Conviene que sigas estos pasos siempre que trabajes en código máquina, ya que si te equivocas copiando los datas de la rutina, puede que pierdas el control del ordenador y tengas que hacer un *reset*. Si te pasa esto y no has guardado el programa, no tendrás más remedio que volver a teclearlo.

Un comentario sobre el listado ensamblador: la rutina es bastante sencilla. Consta de una parte destinada a abrir la ventana, de otra para cerrarla y por último de una pequeña rutina (LEPA) cuya misión es leer los parámetros que pasa el programa BASIC. Tanto en la rutina de abrir como en la de cerrar, después de llamar a LEPA se entra en un bucle controlado por el registro HL (que señala a las direcciones de VRAM) y por los registros D y E, que actúan como contadores de filas y de columnas respectivamente. Este bucle se encarga de ir leyendo uno a uno los bytes que corresponden a la ventana y de trasladarlos a su destino en la VRAM. Un dato interesante es que, para leer o escribir en VRAM, se hace uso de un par de rutinas (RVRAM y WVRAM) de la ROM de **MSX**.

Para ver cómo funciona la rutina tienes que emplear el cargador BASIC. Este, además de cargar la rutina, se emplea para mostrar su funcionamiento. Si te fijas en él, verás que empieza por hacer un CLEAR (en la línea 10) para evitar que haya interferencias entre la rutina y los programas BASIC. Después salta a la subrutina de la línea 1000 que es el cargador en sí y que lleva a la memoria los valores de las sentencias DATA. Fíjate que estos valores, que corresponden a la rutina, son los mismos que los del listado ensamblador. A continuación el programa entra en la zona de carga de parámetros (líneas 12 a 38). Las sentencias REM indican qué parámetro se carga en cada caso. Aquí es donde puedes intervenir tu, modificando los valores que se POKEan en las distintas direcciones, con lo que conseguirás ventanas más anchas, más largas o situadas en otra zona de la pantalla. Concretamente, en la línea 16 puedes elegir el origen de la ventana entre 0 y 959, en la línea 24 puedes variar el número de filas entre 2 y 22 y por último, en la 30 el número de columnas entre 3 y 38. La línea 36 es importante: tienes que poner en ella el valor que se obtiene al restar de 41 el número de columnas que hayas elegido. Por último, y desde la línea 100, el programa se encarga de imprimir algunas frases en pantalla y de dar paso a las distintas rutinas cuando se pulsan las teclas A (Abrir) C (Cerrar) y T (Imprimir Texto). Siempre que tengas el programa guardado en *cassette*, dedícate a hacer todos los experimentos que quieras.

EL MARCO DE LA VENTANA

Si bien hemos logrado abrir y cerrar una ventana, a gran velocidad (ventajas del código máquina), también es cierto que nuestra ventana resulta un poco pobre. En realidad más que una ventana es un simple recorte en la pantalla, sin la menor gracia. Además, cuando la ventana se abre en una zona en la que no hay texto, no es fácil distinguirla ya que queda perdida en el fondo. Así que lo primero que vamos a hacer es ponerle un marco, que delimite claramente lo que es ventana y lo que es pantalla de fondo.

Para ponerle un marco a la ventana vamos a utilizar algunos de los caracteres gráficos **MSX**. Si coges el manual y miras la tabla de caracteres, observarás que los que llevan los códigos hexadecimales 16, 17, 18, 19, 1A y 1B, son idóneos para dibujar un marco. El de código 16 es un borde vertical, el de código 17 es un borde horizontal y el resto son las cuatro esquinas.

¿Cómo incluir estos caracteres en la ventana? Como sabemos dónde empieza la ventana, cuántas columnas tiene y cuántas filas tiene, no tenemos más que preparar una rutina que coloque en las casillas correspondientes los caracteres adecuados. Por ejemplo, la rutina que hemos preparado acude primero a la posición de VRAM que corresponde a la esquina superior izquierda y coloca allí el código 18. A continuación entra en un bucle en el que se imprimen tantos caracteres de borde horizontal como columnas tiene nuestra ventana. Al terminar el borde horizontal, se coloca en el siguiente byte el código 19, que corresponde a la esquina superior derecha. La rutina entra entonces en otro bucle que imprime tantos bordes verticales como filas tiene la ventana menos dos. Las dos filas que faltan son la superior que ya hemos dibujado y la inferior que dibujamos a continuación.

La rutina, como puedes ver en el listado ensamblador (el titulado MARCO) es casi tan larga como la de abrir y cerrar la ventana, aunque lleva a cabo una función en principio mucho más sencilla. Esto es prácticamente

```
                          ;*****MARCO*****
                                ;
                                        ORG   0B06EH
                                        LOAD  0B06EH
                                ;
                          RVRAM:        EQU   4AH
                          WVRAM:        EQU   4DH
                                ;
B06E FD2101B0                           LD    IY,0B001H
B072 FD6E03                             LD    L,(IY+3)
B075 FD6604                             LD    H,(IY+4)
B078 3E18                               LD    A,18H
B07A CD4D00                             CALL  WVRAM
                                    ;
B07D CDBBB0                             CALL  BH
B080 23                                 INC   HL
B081 3E19                               LD    A,19H
B083 CD4D00                             CALL  WVRAM
                                    ;
B086 FD5602                             LD    D,(IY+2)
B089 15                                 DEC   D
B08A 15                                 DEC   D
B08B FD4E00               M1:           LD    C,(IY+0)
B08E 0600                               LD    B,00H
B090 09                                 ADD   HL,BC
B091 3E16                               LD    A,16H
B093 CD4D00                             CALL  WVRAM
B096 FD4E01                             LD    C,(IY+1)
B099 0600                               LD    B,00H
B09B 0D                                 DEC   C
B09C 09                                 ADD   HL,BC
B09D 3E16                               LD    A,16H
B09F CD4D00                             CALL  WVRAM
B0A2 15                                 DEC   D
B0A3 C28BB0                             JP    NZ,M1
                                    ;
B0A6 FD4E00                             LD    C,(IY+0)
B0A9 0600                               LD    B,00H
B0AB 09                                 ADD   HL,BC
B0AC 3E1A                               LD    A,1AH
B0AE CD4D00                             CALL  WVRAM
                                    ;
B0B1 CDBBB0                             CALL  BH
                                    ;
B0B4 23                                 INC   HL
B0B5 3E1B                               LD    A,1BH
B0B7 CD4D00                             CALL  WVRAM
B0BA C9                                 RET
                                    ;
B0BB FD5E01               BH:           LD    E,(IY+1)
B0BE 1D                                 DEC   E
B0BF 1D                                 DEC   E
B0C0 23                   B1:           INC   HL
B0C1 3E17                               LD    A,17H
B0C3 CD4D00                             CALL  WVRAM
B0C6 1D                                 DEC   E
B0C7 C2C0B0                             JP    NZ,B1
B0CA C9                                 RET
                                        END
```

inevitable, ya que se trata de una clásica rutina de formateo en la que hay que incluir una serie de caracteres distintos, en varias posiciones de la pantalla. En la rutina de abrir y cerrar, todo se solucionaba con dos sencillos bucles, uno para las filas y otro para las columnas. Aquí hay que emplear más bucles y además una serie de sentencias sueltas. Desde luego hay otras formas de escribir la rutina. El diagrama de flujo de la rutina es el de la figura 3. Estúdialo y verás como es sumamente sencillo de entender.

¿Cómo acoplamos la rutina? ¿Necesitamos un nuevo cargador y un nuevo programa? La respuesta es no. Vamos a utilizar el programa cargador que ya tenemos y sólo vamos a hacerle unas ligeras modificaciones. En primer lugar, esta rutina para dibujar el marco, vamos a colocarla inmediatamente después de la de abrir y cerrar la ventana. Si comparas los listados en ensamblador de ambas, verás que el de la primera termina en la dirección de memoria &HB06D y que el de la segunda empieza en &HB06E. Si te fijas en la línea 1000 del cargador, verás que los límites del bucle FOR...NEXT corresponden a los de la primera rutina. Para cargar la segunda a continuación no tenemos más que cambiar el valor final del bucle y hacerlo igual a la última dirección de la rutina que es la &HB0CA.

Además de modificar el contador del bucle, hay que añadir las sentencias DATA con los valores de la rutina. También hay que incluir una sentencia (la 11) en la que se defina la dirección de entrada de la nueva rutina.

Por último, en cuanto al uso de la rutina, no hay que darle demasiadas vueltas. Parece lo lógico dibujar el marco inmediatamente después de haber abierto la ventana. Así que lo que vamos a hacer es: siempre que llamemos a la rutina de abrir la ventana, llamaremos a continuación a la de dibujar el marco. Esto nos obliga a modificar únicamente la línea 130.

Todas estas modificaciones quedan reflejadas en el siguiente listado, que tienes que añadir al programa cargador. Cuando lo hayas tecleado, comprobado y almacenado, escribe RUN

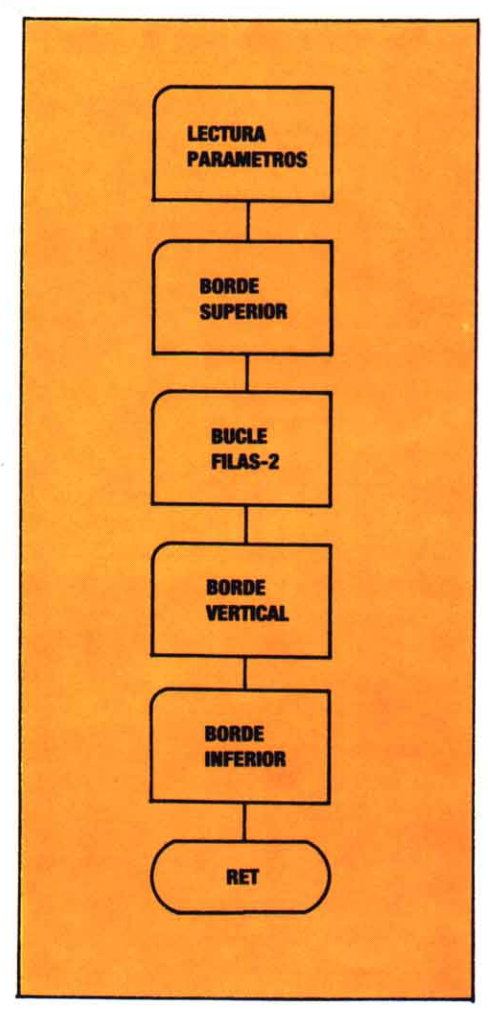


FIGURA 3

y prueba a abrir la ventana igual que antes.

```
11 DEF USR2=&HB06E
130 IF T$="A" THEN U=USR0(0)
    :U=USR2(0)
1000 FOR N=&HB006 TO &HB0CA
2008 'MARCO
2010 DATA FD,21,01,B0,FD,6E,
     03,FD,66,04,3E,18,CD,4D,
     00,CD,BB,B0,23,3E,19,CD,
     4D,00,FD,56,02,15,15,FD,
     4E,00,06,00
2020 DATA 09,3E,16,CD,4D,00,
     FD,4E,01,06,00,0D,09,3E,
     16,CD,4D,00,15,C2,8B,B0,
     FD,4E,00,06,00,09,3E,1A,
     CD,4D,00
2030 DATA CD,BB,B0,23,3E,1B,
     CD,4D,00,C9,FD,5E,01,1D,
     1D,23,3E,17,CD,4D,00,1D,
     C2,C0,B0,C9
```

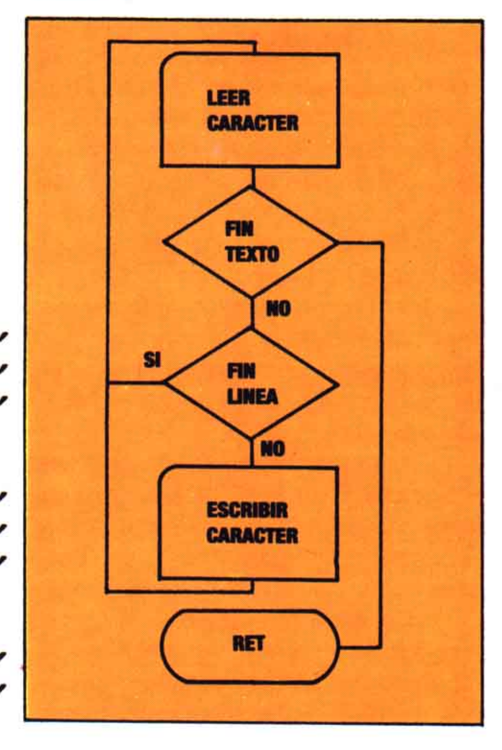
TEXTO PARA LA VENTANA

El paso final en este viaje a través de las ventanas, va a consistir en una rutina para incluir cualquier texto dentro de la ventana. El texto habrá que incluirlo en el programa BASIC que haga uso de las ventans, mediante unas cuantas sentencias DATA. En este caso también se da una comunicación entre el programa BASIC y la rutina en código máquina. El programa BASIC se encargará de POKEar los códigos ASCII de cada uno de los caracteres que componen el texto, además de los códigos de un par de caracteres de control que utilizaremos para señalar el fin de línea y el fin de texto. Todos estos caracteres se POKEarán a partir de la dirección &HB200, un poco más allá del final de la rutina de ventanas.

Esta, cuando entre en acción, se encargará de acudir a la dirección &HB200 y siguientes, irá leyendo los caracteres que dejó el programa BASIC y los irá imprimiendo en pantalla, dentro de la ventana.

A la vista del diagrama de flujo de la rutina, (figura 4) conviene comentar un par de cosas. En primer lugar,

FIGURA 4



```
                  ;*****IMPRESION TEXTO*****
                  ;
                          ORG  0B0CBH
                          LOAD 0B0CBH
                  ;
                  RVRAM:  EQU  4AH
                  WVRAM:  EQU  4DH
                  ;
B0CB FD2101B0             LD   IY,0B001H
B0CF DD2100B2             LD   IX,0B200H
B0D3 FD6E03               LD   L,(IY+3)
B0D6 FD6604               LD   H,(IY+4)
B0D9 1600                 LD   D,0
                      ;
B0DB 0E29                 LD   C,41
B0DD 0600                 LD   B,00H
B0DF 09                   ADD  HL,BC
                      ;
B0E0 DD7E00       IMPRE:  LD   A,(IX+0)
B0E3 DD23                 INC  IX
B0E5 FE40                 CP   40H
B0E7 C8                   RET  Z
B0E8 FE23                 CP   23H
B0EA CAF5B0               JP   Z,FINLIN
B0ED CD4D00               CALL WVRAM
B0F0 23                   INC  HL
B0F1 14                   INC  D
B0F2 C3E0B0               JP   IMPRE
B0F5 4A           FINLIN: LD   C,D
B0F6 0600                 LD   B,0
B0F8 37                   SCF
B0F9 3F                   CCF
B0FA ED42                 SBC  HL,BC
B0FC 0E28                 LD   C,40
B0FE 0600                 LD   B,0
B100 09                   ADD  HL,BC
B101 1600                 LD   D,0
B103 C3E0B0               JP   IMPRE
                          END
```

la rutina en código máquina no mide la longitud de los textos que imprime. Esto quiere decir que si alguna de las líneas de texto es más larga que el ancho de la ventana, se saldrá de ésta. Esto obliga a no incluir ninguna línea de texto que tenga más caracteres que el ancho de ventana elegido. Es decir, si elegimos una ventana de 15 columnas e incluimos una línea de texto con más de 15 caracteres, nos encontraremos con que la línea se sale de la ventana.

Otra cuestión es la de los caracteres de control. Para indicarle a la rutina el fin de una línea, utilizaremos el caracter #, mientras que para indicarle el final del texto usaremos el caracter @. Si olvidamos algunos de estos caracteres, ocurrirán cosas bastante extrañas y puede ser que nos quedemos colgados, sin control sobre el ordenador.

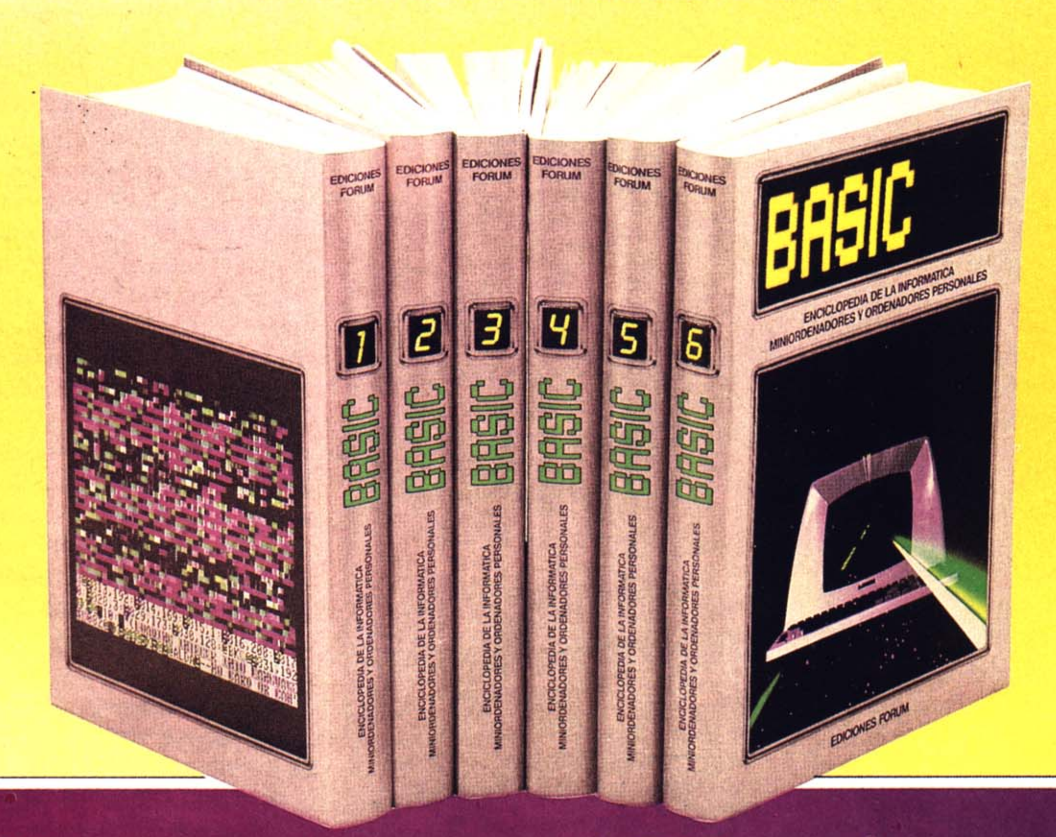
Todo esto se ve bien claro en el ejemplo incluido en el cargador. Modifica las líneas del programa que tienes hasta ahora con lo que sigue:

```
10 CLEAR 200,&HB000:GOSUB
   1000:GOSUB3000:DEF
   USR0=&HB006:DEF
   USR1=&HB035
11 DEF USR2=&HB06E
   :DEF USR3=&HB0CB
130 IF T$="A" THEN U=USR(0)
    :U=USR2(0):U=USR3(0)
1000 FOR N=&HB006 TO &HB105
2040 'TEXTO
2042 DATA FD,21,01,B0,DD,21,
     00,B2,FD,6E,03,FD,66,04,
     16,00,0E,29,06,00,09,DD,
     7E,00,DD,23,FE,40,C8,FE,
     23,CA,F5,B0
2044 DATA CD,4D,00,23,14,C3,
     E0,B0,4A,06,00,37,3F,ED,
     42,0E,28,06,00,09,16,00,
     C3,E0,B0
3000 DIR=&HB1FF
3002 READ CA$
3004 IF CA$="FIN"
     THEN RETURN
3006 FOR J=1 TO LEN(CA$)
3008 LE$=MID$(CA$,J,1)
3010 POKE DIR+J,ASC(LE$)
3012 NEXT J
3014 DIR=DIR+LEN(CA$)
3016 GOTO 3002
4000 DATA 1.-DATOS#,2.-LISTAR
     #,3.-SALIR#,4.-BASIC#,#,
     #,ELIGE@,FIN
```

El salto a la subrutina 3000, en la línea 10, da paso a la lectura de los textos. Estos son los que aparecen en las sentencias DATA de la líneas 4000 y siguientes. Observa bien la estructura, los caracteres de fin de línea y el de fin de texto. No olvides incluirlos cuando escribas tus propios textos. La línea 11 incluye ahora la definición de la nueva subrutina a la que llamaremos con USR3(0). No te olvides de modificar los límites del bucle cargador (línea 1000). Por último, en la línea 130 se hace la llamada a la rutina de impresión de textos. Cuando la utilices en tus programas conviene que la llames siempre después de haber abierto la ventana y haber dibujado el marco, es decir, utiliza la secuencia USR0, USR2, USR3 tal y como lo hace el ejemplo.



ENCICLOPEDIA DE LA INFORMATICA DE LOS MINIORDENADORES Y ORDENADORES PERSONALES



84 fascículos semanales de 24 páginas cada uno.

6 volúmenes de gran formato (19,5 × 27,5) encuadernados en geltex impreso a todo color.

1.748 páginas en papel especial.

2.000 gráficos e ilustraciones a color.

BASIC

Una información indispensable del primero al último fascículo.

BASIC

Para no ser un extraño en el futuro tecnológico que nos aguarda.

BASIC

Para poner una nueva ciencia a nuestro servicio.

Con su primer fascículo recibirá GRATIS el fascículo n.º 2, es decir, su primer envío constará de 5 fascículos al precio de 4.

SUPER OFERTA DE LANZAMIENTO

RECORTE ESTE CUPON Y ENVIELO A EDISA (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid

SI, deseo suscribirme a **BASIC** y recibiré en mi hogar 4 fascículos al mes, abonando sólo **700 Ptas.** por cada envío. El servicio a mi domicilio es GRATUITO.

▽ POR FAVOR, RELLENE SUS DATOS EN MAYUSCULAS ▽

NOMBRE ______

APELLIDOS ______

DIRECCION ______ N.º ______

PISO ______ COD. POSTAL ______

CIUDAD ______

PROVINCIA ______

TLFNO. ______

FIRMA ______

PROGRAMANDO AVENTURAS

- QUE ES UN JUEGO DE AVENTURAS
- COMO JUGARLO
- SUGERENCIAS PARA
- RESOLVER AVENTURAS
- ESCRIBE TU PROPIO JUEGO

Transporta a tus amigos a un mundo de fantasía de tu propia creación, y proporciónales algún quebradero de cabeza. Nos asomamos al mundo y a la historia de los juegos de aventuras.

Para los que queráis descansar un poco de los juegos de marcianitos con disparos, exite una alternativa: los juegos de aventuras. En este tipo de juego el participante se ve totalmente inmerso en un mundo de fantasía creado por el programador. Ejercitando su buen juicio, su inteligencia y su conocimiento de los hechos y personajes raros que encuentre, viaja por un mundo de fantasía intentando completar la búsqueda imaginada por el programador.

En los próximos números de **INPUT MSX** aprenderás la manera de escribir tus propios juegos de aventuras, pero primero veremos una introducción a estos juegos y en qué consisten.

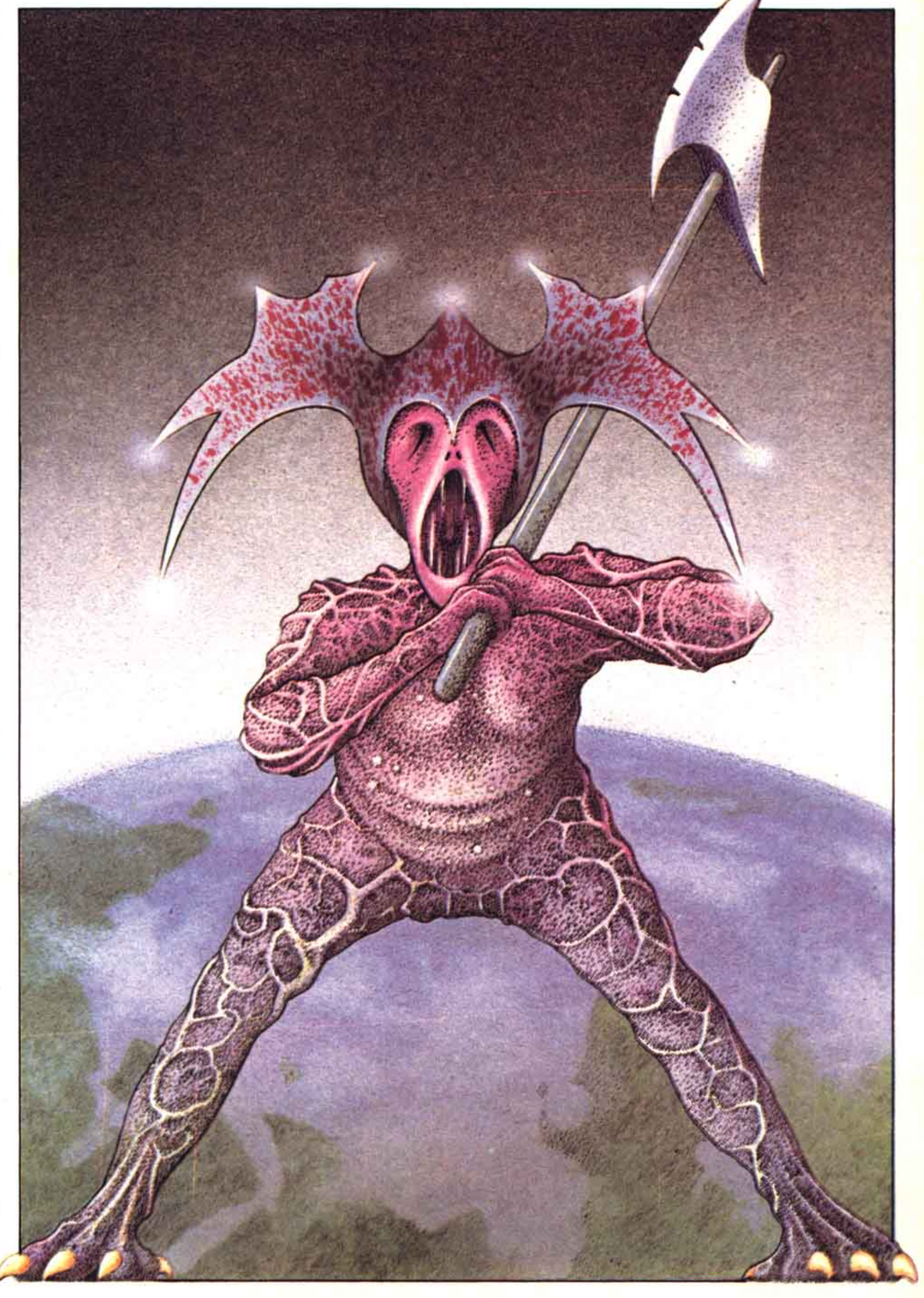
LA HISTORIA DE LAS AVENTURAS

La idea de escribir juegos de aventuras procedía originalmente de la popularidad de juegos de ordenador tales como **Dragones y Mazmorras**, y el deseo de utilizar los ordenadores para algo más que el mero proceso de datos.

En **Dragones y Mazmorras**, el jugador adopta una determinada personalidad y penetra (con su imaginación) en un mundo conocido como la **Mazmorra**, creado por el **Carcelero**. En los juegos de aventuras el programador adopta un papel similar al del **Carcelero**, creando un mundo propio. Por su parte el jugador juega un papel semejante al de alguno de los personajes del juego.

A diferencia del juego tradicional, los jugadores de aventuras por ordenador no pueden elegir normalmente los rasgos de su carácter. De ello se ocupó el programador a la hora de escribir la aventura. En algunas de las versiones más sofisticadas, pueden seleccionar realmente su equipo, etc., antes de empezar su búsqueda. Quizá resulten algo menos sedientos de sangre, aunque naturalmente esto se deja al criterio del autor.

La primera aventura se escribió en un gran ordenador, y no se utilizó el BASIC sino el FOR-



TRAN. El programa ocupaba 300K de memoria, algo más que lo que lleva incorporado tu microordenador.

Sin embargo, el verdadero principio con los micros fué debido a **Scott Adams**, que trasladó algunas ideas al célebre **TRS 80** en 1978, demostrando que era totalmente factible escribir un juego de aventuras satisfactorio precisando menos espacio de memoria. Desde entonces los temas de aventuras que **Adams** adaptó para sus juegos **—Aventurlandia**, **La Cueva del Pirata**, **El Misterio de la Casa Encantada—** han sido utilizados múltiples veces.

TIPOS DE AVENTURAS

Tanto el juego original para un gran ordenador, como los juegos de **Adams** para microordenador únicamente presentan texto sobre la pantalla. Este tipo de aventuras, en las que no aparece ningún tipo de gráfico, sólo textos, siguen siendo las más populares, y hay quien afirma que son los mejores tipos de aventura.

Las aventuras de texto existen realmente en la mente del jugador.Cuando juegues con una de estas buenas aventuras de texto, te verás totalmente envuelto en la historia.

Los gráficos tienen que ser muy sofisticados para poder competir con tu imaginación. Por ejemplo, es posible que te imagines un ogro mucho más feroz que cualquiera que pueda salir incluso de la mejor de las pantallas gráficas, por lo que es muy posible que los gráficos echen a perder tu disfrute. Otra importante consideración en contra de los gráficos es que la pantalla requiere una gran cantidad de memoria, que en otro caso podría servir para alargar más la aventura. Además es posible que resulte demasiado lenta, debido a que el jugador tiene que esperar a que se dibuje una nueva imagen cada vez que se cambia de situación.

Algunas aventuras dan cierta puntuación al completar determinadas etapas, de forma que si te matan en algunas puedas juzgar lo bien o mal que lo has hecho. Algunas incluso te dan una categoría, por ejemplo novato o experto. En el otro extremo de la escala están las aventuras en que no se te da ninguna clave sobre si lo estás haciendo bien o mal, o cuánto te falta hasta la meta final. La última satisfación procede del hecho de resolver una serie de rompecabezas sin fin y de ir acercándose cada vez más hasta encontrar el final del asunto y resolver la aventura.

JUGANDO A LAS AVENTURAS

Cuando ejecutas una aventura, normalmente el programa te dice algo acerca del mundo en el que te vas a encontrar, puede ser en algún paraje exótico de la Tierra, un planeta de una galaxia lejana o en un mundo sólo existente en la fantasía. El juego puede desarrollarse en el pasado, el presente o el futuro, o incluso en una mezcla de los tres. Normalmente te dará unas cuantas indicaciones informativas de base que te servirán de ayuda, tales como quién manda en ese mundo, quién eres tú (si has asumido una determinada personalidad), algo sobre tus amigos y enemigos, y lo que es más importante, lo que tienes que hacer para resolver la aventura y ganar el juego. Lee con cuidado las instrucciones, ya que normalmente contienen mucha información importante.

Después de todo esto, aparecerá la primera descripción del lugar. Probablemente te dirá algo como esto:

TE ENCUENTRAS CERCA
DE UNA ENORME
OLLA LLENA HASTA LOS
BORDES DE UN
ESPUMEANTE
LIQUIDO VERDE.
HAY UN OLOR MALIGNO
EN EL AMBIENTE.
EN EL SUELO
HAY UNA GRAN
CUCHARA.
PUEDES IR HACIA
EL ESTE,
OESTE, NORTE

¿QUE HACES AHORA?

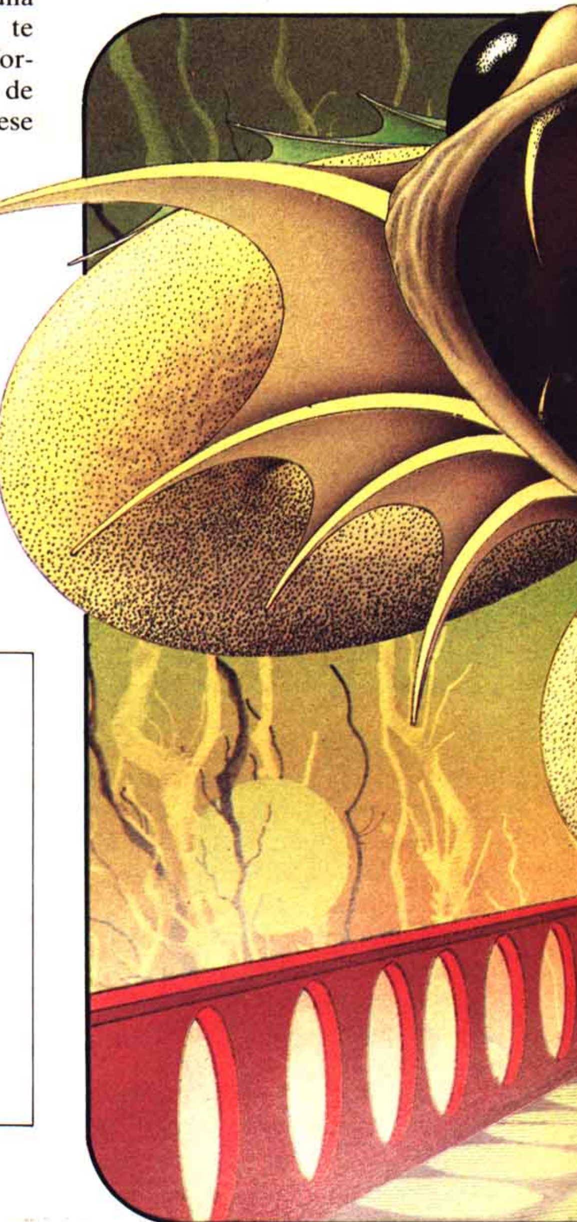
Tienes que decidir lo que quieres hacer. ¿Usarás la cuchara para remover el líquido, o incluso para intentar beber algo? ¿La dejarás ahí? ¿O te dedicarás a explorar, buscando una botella o algún otro recipiente para poder llevarte un poco de líquido verde?

Si te decides a usar la cuchara, tendrás que teclear algo así:

COGER CUCHARA

a lo que el ordenador replicará OK (muy bien), o tal vez NO PUEDES COGER CUCHARA, TODAVIA!, o cualquier otro mensaje.

En cada etapa del juego tienes que decirle al ordenador exactamente lo que quieres hacer, cómo se lo dices



depende del juego. Casi todos los juegos esperarán que comuniques tus instrucciones al ordenador con un verbo en infinitivo seguido de un nombre, por ejemplo, COGER CUCHARA, ESTRANGULAR ELEFANTE, ARRANCAR ARBOL, etc.

Otros juegos más sofisticados aceptarán frases completas, pero esto es más bien la excepción que la regla. Dicha clase de juegos permite decir algo como MATA A ESE PESADO INSECTO DANDOLE UN PISOTON MIENTRAS CANTAS «ALL YOU NEED IS LOVE» (popular canción de los **Beatles**). Un programa que acepte instrucciones tan complicadas como ésta, tendrá que ser forzosamente muy complejo y está fuera de los objetivos de un principiante.

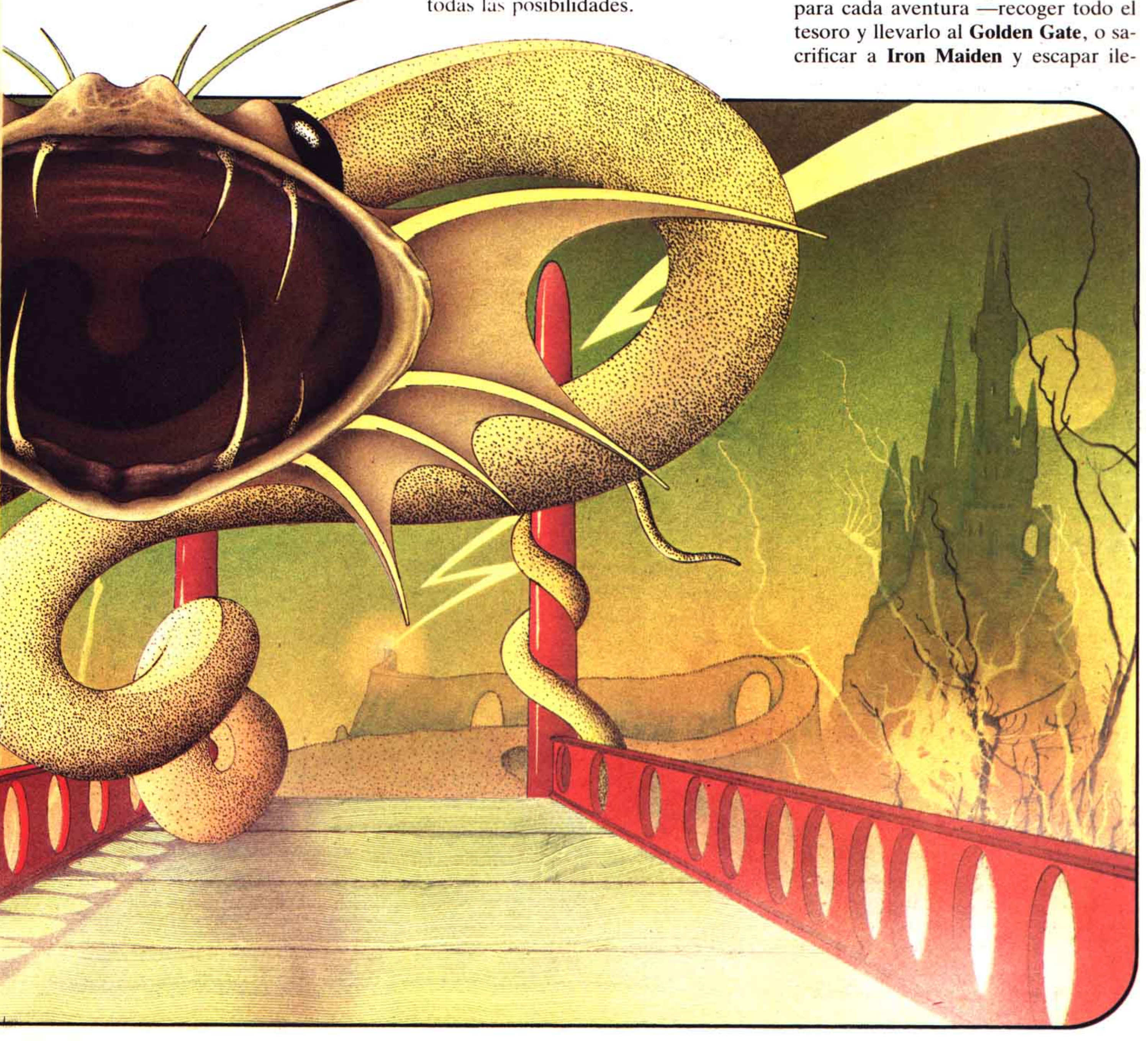
La mayoría de los juegos de aventuras entenderán e incluso esperarán versiones abreviadas de las palabras. Por ejemplo, en los juegos de aventuras es muy normal teclear N en lugar de Norte. Utilizando este tipo de abreviaturas puedes agilizar el juego y ahorrar espacio de memoria.

Las direcciones pueden no ser precisamente N, S, E y O. Podrías encontrarte con ARRIBA o ABAJO, o incluso con NE, SE, SO y NO. Si el juego no te dice qué direcciones tienes disponibles, no te olvides de probar todas las posibilidades.

El caso más normal es que el mundo de la aventura esté basado en una retícula de posibles lugares, habitualmente un cuadrado. Lo que puedan representar estos lugares queda a capricho del programador, pueden ser las estancias de un castillo, o las cámaras subterráneas de una mina. El eslabón de unión entre distintos lugares puede ser algo obvio, como una puerta o un tramo de escaleras, o puede estar menos claro, por ejemplo un río que tienes que atravesar a nado.

RESOLVIENDO AVENTURAS

Normalmente sólo hay una solución para cada aventura —recoger todo el tesoro y llevarlo al **Golden Gate**, o sacrificar a **Iron Maiden** y escapar ile-



so— y una secuencia fija de problemas que resolver. Lo más probable es que necesites muchos, muchos intentos para resolver el juego antes de que termine la aventura. De hecho toda aventura que no te requiera días, semanas o incluso meses de sudores, no es muy buena.

Existen algunas reglas y sugerencias básicas que te ayudarán a resolver un poco más rápidamente la mayoría de las aventuras.

Casi sin excepción, todos los objetos que encuentres en las aventuras tienen algún uso. Para el programador supondría un gasto de memoria innecesario llenar todo de objetos que luego nadie va a utilizar, pero ten cuidado: algunos objetos podrían ser «armas de doble filo». Por ejemplo, es posible que necesites llevar una bolsa con monedas de oro para pasar un puente de peaje, pero si te decides a pasar el río a nado, su peso podría hacer que te hundieras. En general, coge todos los objetos que puedas, aunque a veces te encontrarás que sólo puedes acarrear un número determinado de objetos.

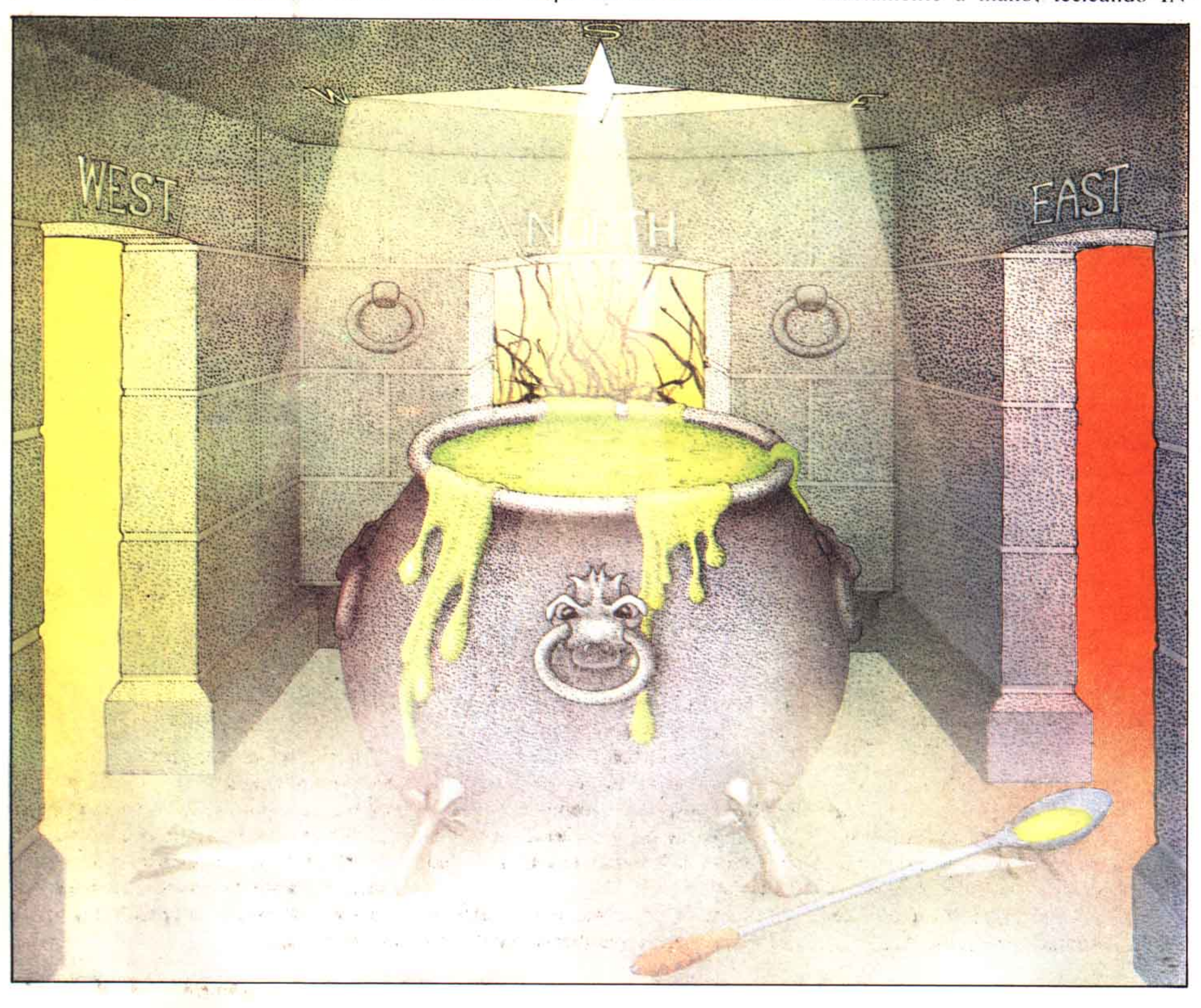
La mayoría de los objetos sólo se utilizan una vez en la aventura. Una excepción podría ser algo como una espada, que se puede utilizar muchas veces para luchar contra los malvados. Si tienes limitado el número de objetos que puedes acarrear, recuerda que lo más seguro normalmente es descartar los objetos una vez que los has utilizado.

Dibuja un mapa siempre. Marca sobre el mismo los nombres de las estancias, todo lo que sea de interés sobre cada una de ellas, los objetos que haya dentro, todas las entradas y salidas, y sus direcciones.

El mapa te ahorrará mucho tiempo y esfuerzo cuando tengas que volver sobre tus pasos, cosa que tendrás que hacer muchas veces durante el juego.

Si tienes que abandonar algún objeto debido a que no puedes llevarlo todo, no te olvides de marcar su posición en el mapa. Y lo que también es muy importante, el dibujo del mapa permitirá que te asegures de explorar toda la aventura, con lo que no tendrás que volver a nadar en las arenas movedizas por enésima vez.

Casi todos los juegos te permiten pedir un inventario de los objetos que llevas. De vez en cuando es una buena idea examinar qué objetos tienes exactamente a mano, tecleando IN-



VENTARIO, INVE o simplemente I, dependiendo de la aventura de que se trate.

Algunas aventuras te permiten además pedir ayuda, también esto depende del juego, así como la forma de pedirla. Pudedes conseguir o no una sugerencia útil, lo más frecuente es que te encuentres con algo como AQUI NO HAY AYUDA.

Algunos juegos siguen muy de cerca la descripción de un libro particular, en cuyo caso el estudio del libro en cuestión es decididamente una buena idea. Otros juegos toman prestadas pequeñas secciones o ideas. Si crees reconocer algo y no puedes resolver un problema particular, intenta buscar en el libro. Análogamente si un extraño personaje con un enorme hacha bloquea tu camino y te pregunta el diámetro de la Tierra, no lo dudes, !véte y averígualo!

Otro artificio muy usado en los juegos de aventuras son los equívocos. Míralos bien, no todas las cosas son lo que parecen.

También puede ser una buena idea mantener un directorio de sinónimos, y manejarlo para probar todas las variaciones posibles de una frase particular. Por ejemplo, el programador podría no haber incluído RESTREGAR además de FROTAR.

Y un último truco. Si la aventura que estás jugando te permite guardar (con SAVE) alguna parte, y estás a punto de intentar algo peligroso, guárda la situación en la que te encuentras, antes de probar. Si te matan, simplemente volverás a cargar el juego y podrás continuar desde donde estabas. Con este sistema tendrás muchos intentos para matar al dragón, atravesar un puente que se tambalea o escapar de una caverna.

ESCRIBIENDO AVENTURAS

Escribir aventuras es una buena manera de ponerse seriamente a estudiar BASIC. Se utilizan casi todos los aspectos importantes del lenguaje: manejo de cadenas, las distintas modalidades de PRINT para formateo de pantalla, variables, cadenas, etc.

La mayoría de las aventuras comerciales están aún escritas en BASIC, debido a que realmente no hay necesidad real de aprovechar la velocidad del código máquina. La única barrera para que produzcas juegos de una calidad absolutamente superior es tu propia imaginación.

Sin embargo, antes de sentarte a programar tu aventura, debes tener una idea muy clara de lo que vas a hacer. Si quieres ahorrarte muchos quebraderos de cabeza, debes planear por adelantado los dibujos, los enigmas, los peligros, etc.

Primero coge un papel y anota unas cuantas ideas. No te preocupes si no tienes una visión completa de todo lo que supone un juego de aventuras; lo que necesitas es una idea para una historia, un lugar para la aventura y unos cuantos rompecabezas para que los resuelva el jugador. A medida que vayamos avanzando en esta sección de nuestro coleccionable, verás cómo una idea sobre un juego se va convirtiéndo en una aventura y aprenderás a adaptar tus propias ideas originales.

Has de ser muy cuidadoso con el mundo que elijas. Intenta que sea lo más interesante posible, si pretendes que una aventura resulte muy apasionante en el interior de un bloque de oficinas, te va a costar lo tuyo.

Para tu inspiración puedes acudir a libros, películas, la televisión o cualquier otra posible fuente. También puedes sacar ideas de otros programas de aventuras, aunque probablemente la mejor fuente de inspiración es...!una mente ligeramente retorcida! Busca siempre un tema o idea central que puedas ir desarrollando por toda la aventura.

Intenta conseguir el adecuado equilibrio entre desafío e imposibilidad. No es bueno gastar mucho tiempo y esfuerzos escribiendo una aventura que cualquiera pueda resolver en media hora. Recíprocamente, no ganarás muchos amigos si tu aventura es totalmente imposible de resolver.

La regla de oro es «dar algunas posibilidades a los jugadores, !pero no demasiadas!»

Intenta no dejar muchas habitaciones vacías en tus aventuras. Realmente no añaden nada a la misma y ocupan un espacio de memoria que es vital. Además contribuyen a que la aventura sea más aburrida.

No hagas que tus primeras aventuras sean muy complejas, ya que los problemas que originen pueden ser muy difíciles de depurar hasta que adquieras cierta práctica. Aprende a conocer todo lo que interviene antes de intentar algo muy ambicioso. Ten a la vista cuánta memoria tiene aún disponible tu máquina.

En la aventura que verás construir más adelante hay muchas sentencias REM. Para ahorrar memoria en una gran aventura es mejor no poner muchas, pero al principio contribuyen a que resulte más fácil de escribir.

También se puede ahorrar memoria en las descripciones de los lugares. No las hagas demasiado cortas, porque podrías perder todo el sabor de la aventura. A ti te corresponde establecer el correcto equilibrio entre sabor y espacio.

En los próximos números veremos cómo convertir una idea sobre una aventura en un mapa y como empezar un programa.

¿CUANTA MEMORIA QUEDA?

Cuando estás escribiendo un gran juego de aventuras, es muy fácil que te encuentres con que has sobrepasado los límites de la memoria de tu máquina. Evidentemente, los problemas de pasarse de capacidad de memoria son más agudos en el caso de máquinas con menor memoria. Hay una forma de comprobar cuánta memoria queda, basta con que utilices la función FRE.

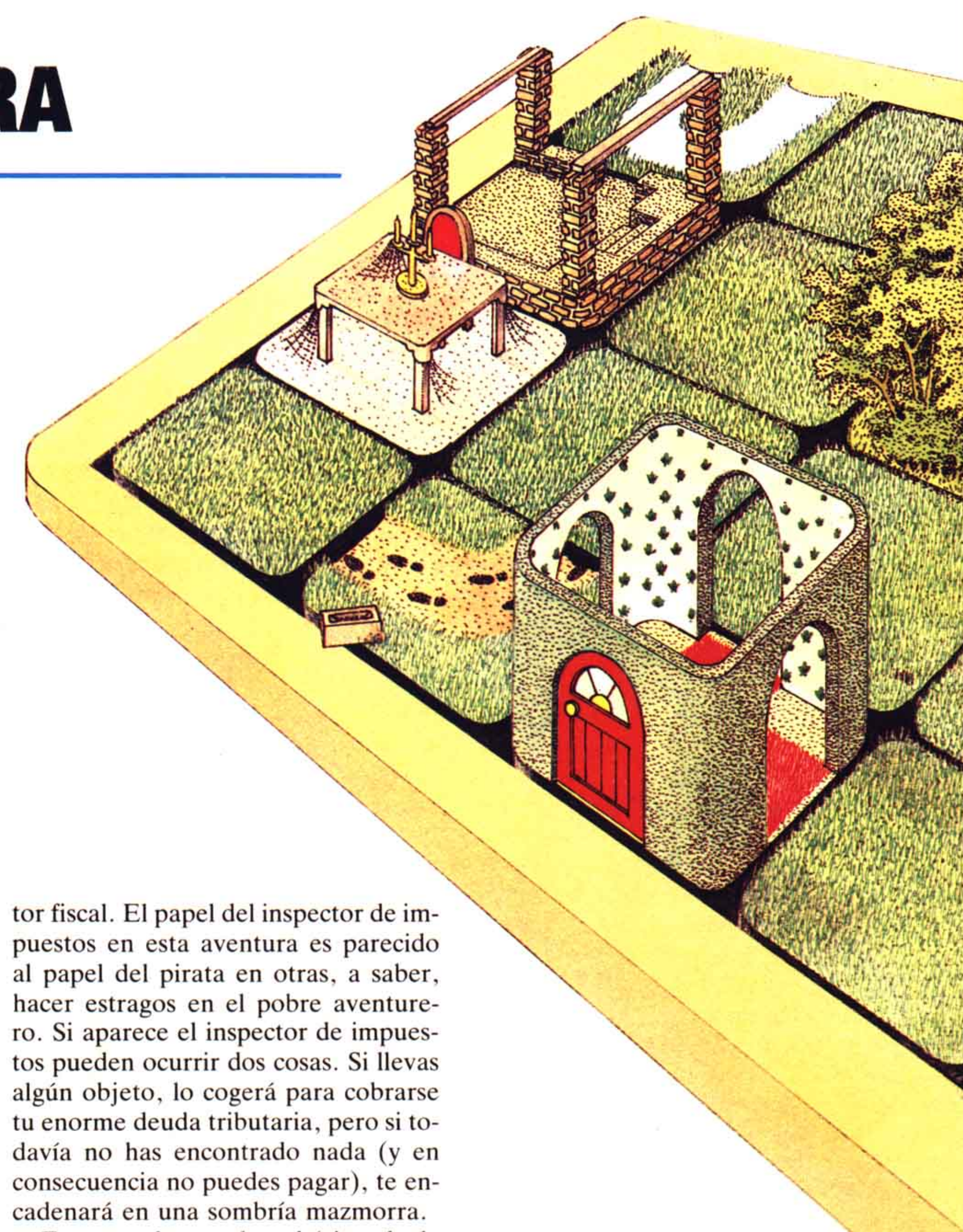
No tienes más que teclear:

```
PRINT FRE(0)
```

La memoria que queda tiene en cuenta tanto el espacio ocupado por el programa como el espacio ocupado por las variables. La mejor manera de encontrar la cantidad exacta de memoria restante es ejecutar el programa (con RUN), si es posible durante el desarrollo.

PROYECTA TU AVENTURA

Al escribir una aventura, la primera etapa es perfilar un bosquejo general de la historia y dibujar un mapa tosco de todos los lugares que intervienen. Esto constituye la base de todo el programa.



Es esencial que tengas toda tu historia lista antes de empezar a programar. Si no lo haces así, es muy probable que tengas muchas dificultades, con muchos errores y cabos sueltos difíciles de atar.

Para ver cómo se hace esto dedicaremos los siguientes capítulos al desarrollo de un programa de aventuras típico (aunque necesariamente muy corto). La acción de la aventura se sitúa en algún lejano país donde el jugador tiene que conquistar el fabuloso ojo perdido de una purpúrea estatua. Si sigues todos los pasos al escribir esta aventura, rápidamente verás la forma de escribir las tuyas propias.

LA HISTORIA

Tienes que crear un mundo que se adapte a las líneas generales de la historia. Has de encontrar objetos adecuados y asignarles un papel, y además tienes que preparar enigmas para ser resueltos.

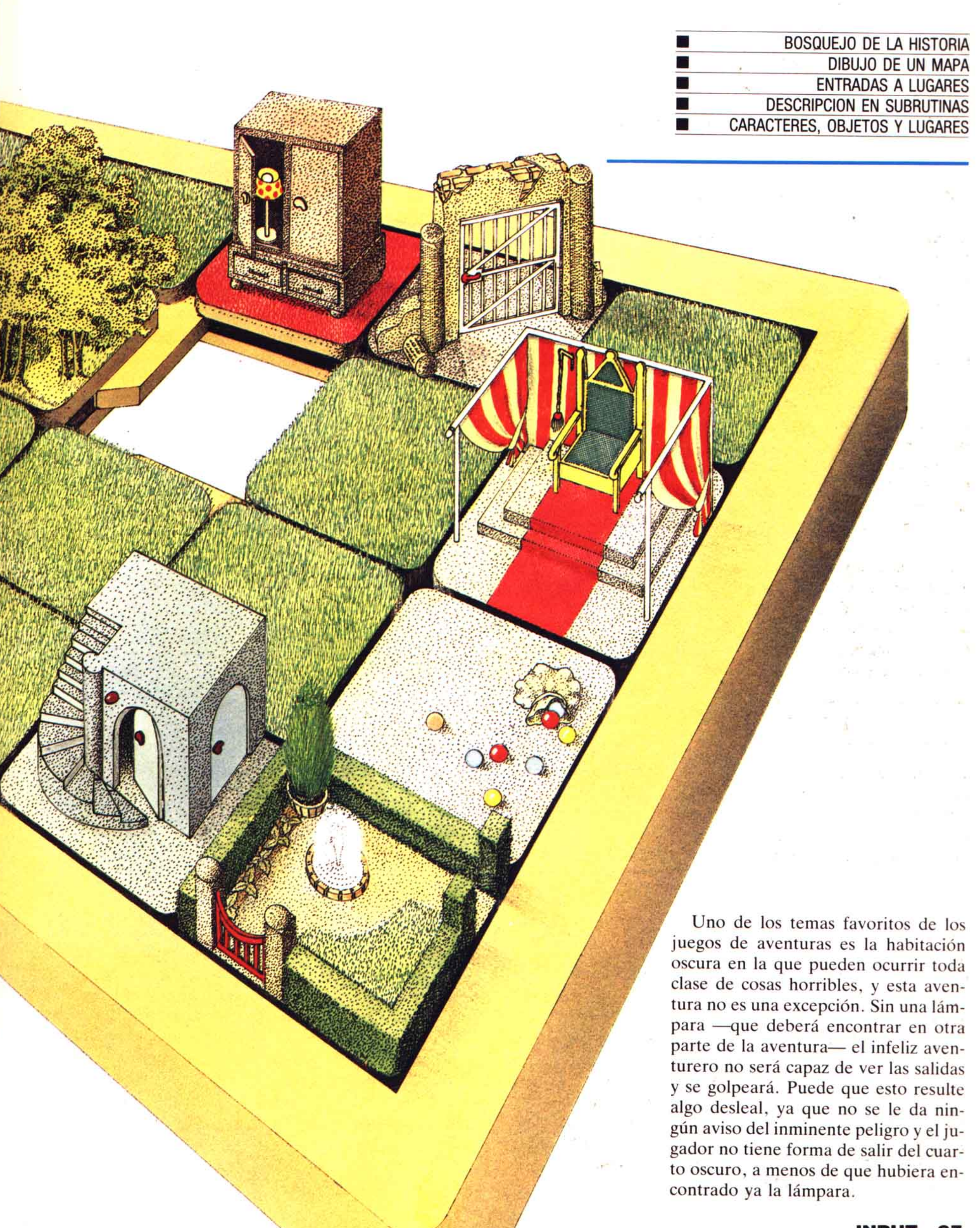
No hace falta que te ocupes de todo esto a la vez, ya que a medida que vas pensando la aventura, la historia va tomando cada vez una forma más definida y los detalles van encajando en su sitio. Empieza pues bosquejando la historia a grandes rasgos.

El jugador de la aventura atraviesa unas dificultades financieras espantosas, se ha embarcado en la búsqueda del fabuloso (y muy valioso) globo ocular que está escondido en alguna parte en el mundo de la aventura. Por desgracia la Delegación de Contribuciones ha enviado tras él a un inspector fiscal. El papel del inspector de impuestos en esta aventura es parecido al papel del pirata en otras, a saber, hacer estragos en el pobre aventurero. Si aparece el inspector de impuestos pueden ocurrir dos cosas. Si llevas algún objeto, lo cogerá para cobrarse tu enorme deuda tributaria, pero si todavía no has encontrado nada (y en consecuencia no puedes pagar), te encadenará en una sombría mazmorra.

Este es el esqueleto básico de la idea. Ahora tienes que ir cubriendo algunos detalles, como qué objetos se pueden encontrar durante la búsqueda. En nuestra aventura, hemos decidido que la regla general de recoger el mayor número posible de objetos no sea válida. Esta vez, no todos los objetos son provechosos para la búsqueda; de hecho uno de ellos no valdrá absolutamente para nada. O para casi nada, es un objeto pesado (por ejemplo un ladrillo) que te matará si intentas cruzar el río a nado.

El objeto más importante de todos es el globo ocular, y para añadir más interés, se ha previsto una manera de esconderlo o enmascararlo. Podría estar escondido dentro de un cofre o en una cámara acorazada, pero hemos elegido una forma mucho más astuta de despistar al aventurero. En vez de ocultar la joya en un sitio que obviamente contiene cosas de valor, estará camuflada en una bolsa de canicas. ¡Cualquier intento de jugar a las canicas, no conducirá al aventurero a ninguna parte!

- BOSQUEJO DE LA HISTORIA
- DIBUJO DE UN MAPA
- ENTRADAS A LUGARES
- DESCRIPCION EN SUBRUTINAS
- CARACTERES, OBJETOS Y LUGARES



Uno de los temas favoritos de los juegos de aventuras es la habitación oscura en la que pueden ocurrir toda clase de cosas horribles, y esta aventura no es una excepción. Sin una lámpara —que deberá encontrar en otra parte de la aventura— el infeliz aventurero no será capaz de ver las salidas y se golpeará. Puede que esto resulte algo desleal, ya que no se le da ningún aviso del inminente peligro y el jugador no tiene forma de salir del cuarto oscuro, a menos de que hubiera encontrado ya la lámpara.

Para neutralizar las acciones del inspector de impuestos, y dar al aventurero alguna probabilidad más, por alguna parte de la aventura se esconderá un arma, tal vez una pistola o un cuchillo.

Finalmente, por pura diversión, hemos puesto un salón del trono y una cadena. El salón del trono no es exactamente lo que parece ser. De hecho, si no llevas la joya, al tirar de una cadena, el agua te arrollará y serás violentamente expulsado de la aventura.

Queda una cosa por establecer, la más importante de todas, las condiciones para ganar el juego. No hay una salida del mundo, y una parte del enigma es cómo escapar con la joya.

Evidentemente, para ganar el juego el aventurero tiene que haber encontrado la joya; no basta con la bolsa de canicas. Para que sea aún más difícil, el aventurero tiene que estar además en el salón del trono. Si tira de la cadena esta vez, no le arrastrará hacia dentro del inodoro.

La ventaja de que la vía de salida sea un peligro en otras condiciones, es que se desanime el jugador cauto y no intente entrar allí muy a menudo, lo cual prolonga el juego.

LA HISTORIA HASTA AHORA

Es el momento de hacer una recapitulación, antes de que pierdas las pistas de los temas, que empiezan a poner ya la cosa un poco complicada. Puede ser útil hacer una lista antes de empezar con el mapa. Para la aventura podría ser algo así:

PERSONAJES:
- Aventurero
- Inspector de hacienda; aparecerá aleatoriamente

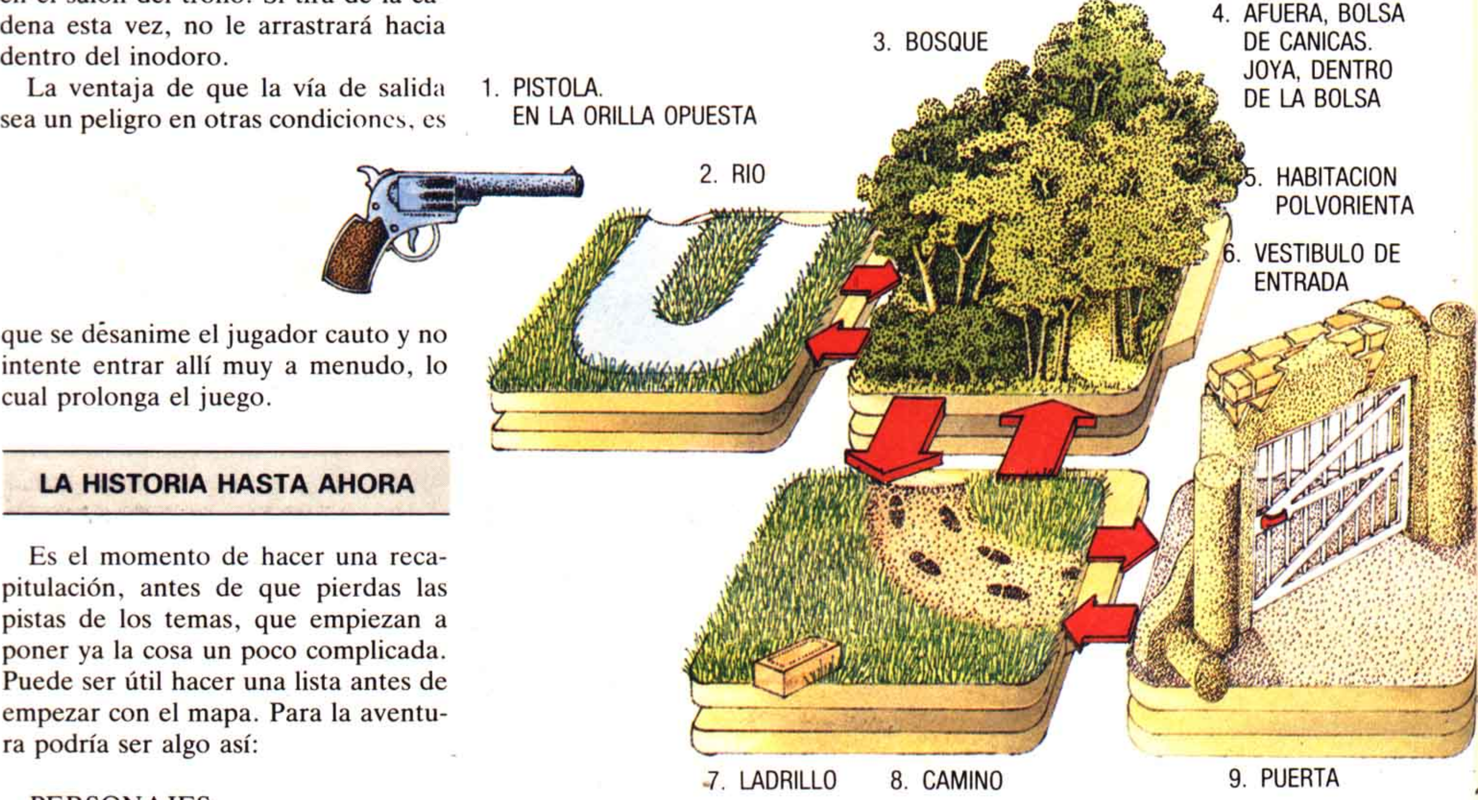
OBJETOS:
- Globo ocular; escondido en una bolsa de canicas
- Ladrillo; motivo de distracción que matará al aventurero si intenta cruzar el río a nado
- Lámpara; necesaria para encontrar la salida de una habitación a oscuras
- Pistola; arma para matar al inspector de impuestos
- Cadena; en el salón del trono

LUGARES:
- Río
- Cuarto oscuro
- Salón del trono

Hasta ahora en la avenura sólo se han fijado tres de los lugares por las cosas que tiene que ocurrir en ellos. En este punto podrías haber decidido algunas cosas más. Pero en cualquier caso, tu siguiente paso es reunir todos estos temas en un plano del mundo de la aventura.

EMPEZANDO CON EL MAPA

Probablemente tu primer mapa consistirá simplemente en una serie de cajas conectadas por medio de flechas, como en la figura 1. Cada una de las cajas representa una habitación o un lugar del mundo; lugar es probablemente el término más adecuado ya que las aventuras no están limitadas a interiores, y lugar puede ser cualquier cosa, desde una cabeza de alfiler en el dobladillo de la reina hasta una enorme llanura que se extiende hasta donde alcance tu vista. Tienes que incorporar todos los lugares en tu lista pre-



El primer mapa de la aventura muestra todos los lugares proyectados y sus posiciones relativas. Las flechas indican salidas que están siempre abiertas, las flechas con rayas indican salidas que sólo se pueden utilizar bajo especiales condiciones; en este caso, cuando se ha encendido la lámpara.



¡NO TE PIERDAS NI UN SOLO EJEMPLAR!

INPUT MSX quiere proporcionar a sus lectores este nuevo servicio de ejemplares atrasados para que no pierdan la oportunidad de tener en sus hogares todos los ejemplares de esta revista, líder en el mercado español.

Podréis solicitar cualquier número de **INPUT MSX** que querais, siempre al precio de cubierta (sin más gastos).

Utiliza el cupón adjunto, enviándolo a **EDISA** (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid, o bien llámanos por teléfono al (91) 415 97 12.





CUPON DE PEDIDO

SI, envíenme contrarreembolso .... ejemplares de **INPUT MSX** de los números:

(marca con una (X) tu elección)

[1] [2] [3] [4]

NOMBRE ____________________

APELLIDOS ____________________

DOMICILIO ____________________

NUM. ____ PISO ____ ESCALERA ____ COD. POSTAL ______

POBLACION ____________________ PROV. ____________

TELEFONO ____________ FIRMA

PREDICIENDO LO IMPREDECIBLE

- LAS REGLAS DEL JUEGO
- SACAR LOS NUMEROS DE
- UNA CAJA O UN SOMBRERO
- NUMEROS ALEATORIOS
- MUESTRAS Y MUESTREO

¿Qué posibilidades tienes de acertar un pleno en las quinielas de esta semana? Con lo que te vamos a contar no es probable que mejore tu suerte, pero al menos tendrás una idea más clara de por qué pierdes.



La instrucción de un novato en el manejo de un moderno avión comercial o de un bombardero sería enormemente cara si hubiera que realizarla sobre aviones reales, debido al costo que suponen factores tales como el combustible, las pistas de aterrizaje y toda la infraestructura de apoyo necesaria. Por eso las autoridades civiles y militares opinan que es más barato gastar dinero en aviones de entrenamiento y simuladores, enormes artefactos controlados por ordenador que nunca abandonan el suelo, pero que hacen que el piloto experimente, con absoluto realismo, la sensación de estar a los mandos de una aeronave.

Este mismo principio resulta cierto en un gran número de circunstancias, incluída la programación de juegos, el desarrollo industrial y el *marketing*, así como la investigación científica y los estudios gubernamentales, sociológicos,etc. Es preferible programar un ordenador para predecir, por ejemplo, el resultado probable de un fallo estructural o las consecuencias una determinada política económica, que realizar el experimento real que podría requerir cinco años de trabajo o incluso podría resultar totalmente imposible de realizar en determinadas circunstancias. No hay que sorprenderse, pues, de que la simulación sea una actividad que despierta cada dia mayor interés y a la que se dedican más y más esfuerzos.

El caso es que, debido a su complejidad, sólo ha llegado a ser verdaderamente rentable con la difusión de los ordenadores. Para llegar a hacer simulaciones interesantes, no es necesario disponer de un equipo superpotente, hasta los micros domésticos, con todas sus limitaciones, pueden realizar un cierto grado de simulación.

Una de las principales herramientas de este tipo de creación de modelos por ordenador es el conjunto de reglas

que nos proporciona el estudio matemático de la probabilidad.

El resto del modelo se deriva de un análisis de la información estadística, obtenido a partir de una descripción de la situación real.

Con unas cuantas reglas sencillas, el usuario del microordenador doméstico puede predecir las alternativas futuras para un sinfín de sucesos diferentes, si bien la fiabilidad de los resultados depende de la precisión de los datos que se introducen en el programa y también de la precisión de las reglas que actúan sobre estos datos. Tan importante son los primeros como las segundas. Si los datos sobre los que trabaja la simulación son datos falsos, el resultado, por muy buenas que sean las reglas que llevan al mismo, será un resultado sin valor. Y lo mismo, aunque trabajemos con datos fiables, si no los manejamos con cuidado podemos extraer de los mismos unas conclusiones equivocadas. En la mayoría de los casos, lo más complejo es llegar a encontrar el conjunto de reglas que permitan extraer conclusiones válidas.

Vamos a ver un pequeño ejemplo de simulación de los resultados de los partidos de futbol.

Todos los fines de semana de la temporada futbolística, millones de aficionados a las quinielas están pendientes de los resultados a la espera de conseguir algún premio sustancioso. Por desgracia, todos menos unos pocos se ven decepcionados y tienen que esperar una semana más para tener otra oportunidad. Si no quieres tener que esperar una semana entera para tener otra oportunidad, teclea el siguiente programa y prueba tu suerte todas las veces que quieras:

```
10 SCREEN0:KEY OFF:WIDTH 40
   :COLOR 15,4,4:CLS
20 DIM A$(14),B$(14),R$(3)
30 LOCATE 8,0
40 PRINT "APUESTA
   FUTBOLISTICA"
50 LOCATE 8,1:PRINT"------- -
   ----------"
100 LOCATE 7,3:PRINT
    "ESCRIBE LOS ENCUENTROS":
    PRINT
110 FOR D=1 TO 100:NEXT D
115 BO$=SPACE$(35)
120 FOR K=1 TO 14
122 LOCATE 1,5+K:PRINT K
125 LOCATE 0,22:PRINT BO$
130 LOCATE 0,22:INPUT
    "EQUIPO LOCAL";A$(K)
135 LOCATE 5,5+K:PRINT A$(K)
140 LOCATE 0,22:PRINT BO$
145 LOCATE 0,22:INPUT
    "EQUIPO VISITANTE";B$(K)
150 LOCATE 22,5+K
    :PRINT B$(K)
160 NEXT K
165 FOR V=1 TO 100:NEXT
170 CLS:LOCATE 8,12:PRINT
    "PARTIDOS DISPUTANDOSE"
180 FOR V=1 TO 1000:NEXT V
190 CLS:LOCATE 7,0:PRINT
    "RESULTADOS DE LA JORNADA
    ":LOCATE 7,1:PRINT "-----
```

```
    ----- -- -- -------"
200 R$(1)="Local":R$(2)=
    "Visitante"
210 R$(3)="Empate"
220 LOCATE 1,3:PRINT
    "ENCUENTRO":LOCATE 20,3
    :PRINT"VICTORIA"
225 R=RND(-TIME)
227 FOR I=1 TO 14
230 Y=RND(1)
240 IF Y<=.5 THEN Z$=R$(1)
250 IF Y>.5 THEN Z$=R$(2)
260 IF Y>.75 AND Y<=.9
    THEN Z$=R$(3)
280 LOCATE 1,4+I:PRINT A$(I);
    "-";B$(I):LOCATE 20,4+I
    :PRINT Z$
290 NEXT I
300 PRINT:PRINT
    "OTRA VEZ (S/N)?"
310 T$=INKEY$:IF T$=""
    THEN 310
320 IF T$="S" OR T$="s"
    THEN RUN
330 IF T$="N" OR T$="n"
    THEN CLS:END
340 GOTO 310
```

Aunque en esta simulación futbolística no hay reparto de dividendos, (incluso aunque hayas acertado una de catorce y cinco de trece), la sensación que se obtiene con ella es bastante realista. Al menos sirve para demostrar lo pequeña que es la probabilidad de conseguir catorce aciertos.

El mal tiempo obliga a veces a suspender determinados partidos. Cuando esto sucede, un grupo de expertos futbolísticos decide qué resultados se habrían producido de haberse jugado realmente el encuentro. De hecho, tanto el programa como el grupo de expertos son simuladores; proporcionan una representación simbólica de un proceso real.

El programa, que cumple las mismas funciones que el grupo de expertos, no es especialmente difícil. Después de introducidos los nombres de los distintos equipos que van a disputar los encuentros de la jornada, el programa entra en la parte principal de simulación, en el bucle comprendido entre las líneas 227 y 290. En él se deciden los resultados de cada uno de los partidos. Una vez decidido cada resultado, la línea 280 se encarga de presentarlo en pantalla.

DECIDIENDO LOS RESULTADOS

La regla en que se basa el ordenador para decidir cualquiera de los resultados, es la vieja regla de las quinielas que dice que la mitad de los resultados son victorias domésticas, y que el resto lo componen las victorias fuera de casa y los empates. Esta regla, aunque discutible, parece proporcionar unos resultados que se acercan bastante a la realidad. En las líneas 240, 250, y 260 se establecen las condiciones que acabamos de comentar. Si tienes tus ideas propias y quieres cambiar las proporciones, para que, por ejemplo, sean todavía más probables las victorias en casa, sólo tienes que modificar los límites que aparecen en las comparaciones. De todas formas conviene que tengas claro que, aunque los resultados del programa pueden acercarse mucho a la realidad, en cuanto a la proporción de cada uno de los resultados, es practicamente imposible que acierten con los equipos correctos. Es más, la simulación no tiene en cuenta, absolutamente para nada, el nombre de los equipos. Se limita a generar unos resultados aleatorios, que eso sí, aparecen en una determinada proporción. Si quieres puedes intentar modificar el programa para que tenga en cuenta aspectos como: el historial de los equipos, las posibles lesiones de los jugadores, los resultados que se han producido en los enfrentamientos de los últimos años, los traspasos de figuras, las condiciones de los campos en los que se juegan los partidos, etc. La idea es la de aumentar las probabilidades de victoria de un equipo cuando se den circunstancias favorables, y a la inversa, disminuir sus probabilidades de victoria cuando, por ejemplo, tenga muchos jugadores lesionados. Con todo ello, y estudiando cuidadosamente las reglas que aplicas, es posible que llegues a un interesante programa de simulación de las jornadas de liga que te permita acertar más quinielas de las que ahora aciertas. El tema es lo suficientemente complejo como para que te pases horas programando y no consigas resultados. Pero a lo mejor te merece la pena intentarlo.



GENERACION DE NUMEROS ALEATORIOS

Hace años se utilizaban métodos manuales o mecánicos, tales como lanzar dados, barajar cartas o emplear una ruleta, para generar listas de números aleatorios. Estos procedimientos eran lentos y tediosos, por lo que el famoso matemático norteamericano



John Von Neuman propuso la técnica del cuadrado medio. Se empieza con un número de cuatro cifras (la semilla), el siguiente número aleatorio se obtiene multiplicando la semilla por sí misma y quedándose con los cuatro dígitos centrales. Por ejemplo, supongamos que la semilla es 5272. El segundo número se genera tomando los cuatro dígitos centrales de (5272^2) que es 27 793 984. El resultado (7939) es prácticamente aleatorio. Se puede obtener un segundo número elevando al cuadrado el número 7939, continuando así el proceso.

Naturalmente, surge la pregunta: ¿Cómo puede un proceso matemático, que es esencialmente repetitivo, producir números verdaderamente aleatorios? La respuesta es que en realidad no puede hacerlo. Sin embargo los números así obtenidos se comportan como si realmente fueran aleatorios, por lo que generalmente se les llama pseudoaleatorios o quasi-aleatorios. De hecho los números generados utilizando la función RND son pseudoaleatorios. Lástima que esta técnica del cuadrado central no sea muy útil para la generación de números aleatorios por ordenador.

Aparte de ser lenta, la secuencia se repite enseguida y en cuanto se obtiene un cero se repite todo el proceso (al elevar cero al cuadrado el resultado sigue siendo cero, y al extraer los cuatro digitos centrales seguimos teniendo cero). La mayoría de los micros domésticos utilizan un método de **congruencias**, que utiliza los restos para generar secuencias pseudoaleatorias. El siguiente programa utiliza una sencilla fórmula y la función INT (parte entera) para proporcionar un ejemplo de esta técnica:

```
20 CLS:KEY OFF:WIDTH 40
30 LOCATE 5,0:PRINT"NUMEROS
   PSEUDOALEATORIOS"
40 LOCATE 2,22:INPUT
   "CUANTOS NUMEROS";N:S=0
   :LOCATE 2,22:PRINT
   SPACE$(35)
50 R=RND(-TIME)
   :R=4*RND(1)
   :X=.677829*R
55 LOCATE 0,4
60 X=X*1842.95
70 X=X-INT(X)
80 P=INT(X*1000)
   :PRINT"      ";1E-03*P,
90 S=S+1
110 IF S<=N THEN
    GOTO 60
120 LOCATE 2,22:PRINT
    "OTRA SERIE (S/N)?"
130 T$=INKEY$:IF T$=""
    THEN 130
135 IF T$="S" OR T$="s"
    THEN RUN
140 IF T$="N" OR T$="n"
    THEN CLS:END
145 GOTO 130
```

La línea 50 utiliza la función TIME del BASIC para establecer como punto de partida un número diferente en cada ejecución. El número .677829 es una constante arbitraria. Después de multiplicar el valor de partida por otra constante (línea 60), se obtiene el resto de la parte decimal (línea 70). Cambia el valor de las constantes de las líneas 50 y 60 y ejecuta nuevamente el programa para ver qué tipo de resultados obtienes.

En muchos casos es preferible —y bastante más sencillo— utilizar la función RND que lleva incorporada tu ordenador. Al modificar el valor de x en la expresión RND(x) podrás seleccionar distintas secuencias.

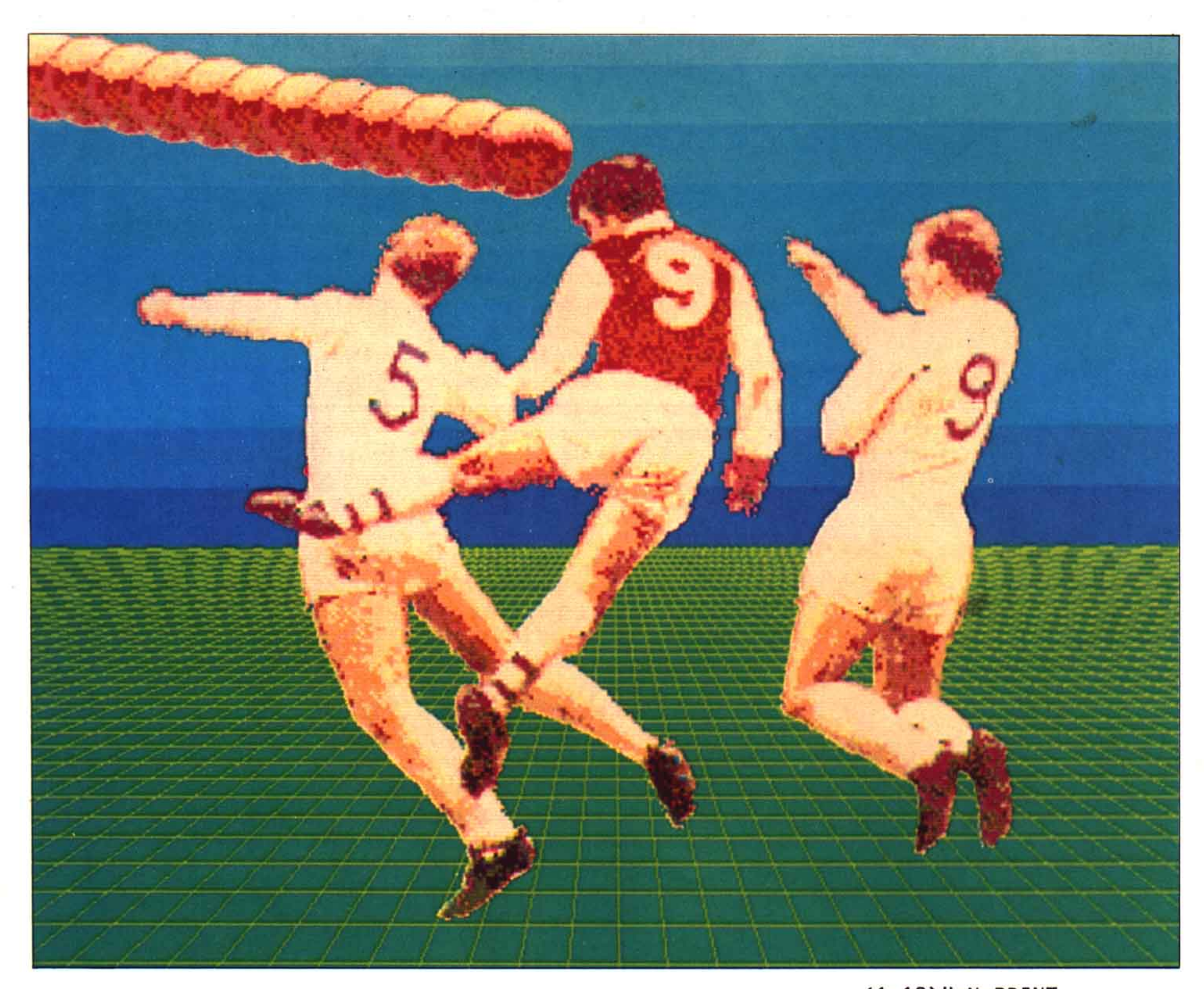
Quizá el punto más importante que tienes que recordar cuando escribas programas en los que utilices la función RND, es inicializar dicha función utilizando la sentencia X=RND(−TIME). Si no lo haces, cada vez que ejecutes el programa obtendrás la misma serie de números aleatorios. Prueba con el programa que acabas de teclear. Suprime la línea 50, hasta los primeros dos puntos, y prueba a ver que pasa.

MUESTRAS Y OBSERVACIONES

Los sondeos preelectorales, las organizaciones de consumidores y los gobiernos utilizan los ordenadores para generar muestras aleatorias de población.

Por ejemplo, es importante que una empresa dedicada a la investigación de mercados que entrevista a 1000 personas, obtenga una muestra típicamente variada, que pueda representar a un grupo mucho mayor de población. No sería válido por ejemplo elegir a todos los individuos entre los miembros de un club de coches antiguos, cuando lo que se pretende investigar son las costumbres nacionales en materia de preferencias automovilísticas.

La mejor forma de hacer que una muestra sea representativa es obtenerla aleatoriamente, intentando evitar cualquier tipo de sesgo o tendencia. Esto sigue siendo cierto también cuando se trabaja con una muestra más pe-



queña, como podría ser el caso de un club de amigos de los coches antiguos o de los lectores de **INPUT**. También en este caso es importante la selección aleatoria, por ejemplo en el caso en que quisieras determinar las preferencias de los lectores de las revistas de informática, no sería lógico que entrevistaras sólo a lectores de Madrid. Las preferencias podrían ser distintas en Barcelona, en Sevilla o en cualquier otra ciudad, y los resultados que obtuvieras tendrían un cierto sesgo, no serían representativos de todos los lectores.

También puede ser que quieras extraer una muestra de tu propia lista de miembros de un club informático. El proceso de muestreo es básicamente similar a la simulación de una extracción de papeletas de un sombrero. No obstante en la simulación, cada papeleta vuelve a ser introducida en el sombrero antes de proceder a la siguiente extracción. En el muestreo, cada papeleta seleccionada se deja fuera del sombrero y se realiza la siguiente extracción. Teclea el siguiente programa para ver cómo se utiliza RND para generar una muestra aleatoria:

```
10 WIDTH40:KEY OFF
   :CLS:R=RND(-TIME)
20 DIM B$(10),A$(10)
30 LOCATE 9,0:PRINT
   "MUESTREO ALEATORIO"
50 LOCATE 9,1:PRINT
   "-------- ---------"
90 FOR J=1 TO 200:NEXT J
100 FOR I=1 TO 10
   :READ A$(I):NEXT I
110 LOCATE 0,4:INPUT
   "MEDIDA DE LA MUESTRA
   (1-10)";N:PRINT
120 FOR V=1 TO 10:B$(V)=A$(V)
   :NEXT V
130 FOR J=1 TO N
140 R=1+INT(RND(1)*10)
150 IF B$(R)="" THEN
   GOTO 140
160 PRINT B$(R)
170 B$(R)=""
180 NEXT J
190 LOCATE 0,22:PRINT
   "OTRA MUESTRA (S/N)?"
195 T$=INKEY$:IF T$=""
   THEN 195
200 IF T$="S" OR T$="s"
   THEN RUN
210 IF T$<>"N" AND T$<>"n"
   THEN GOTO 195
215 CLS:END
220 DATA BONN,COPENAGUE
   ,ESTOCOLMO
230 DATA LONDRES,MADRID
```

```
    ,MOSCU
240 DATA NUEVA YORK,PARIS
    ,ROMA,VIENA
```

Después de leer los datos (línea 100) se genera un número entero aleatorio R comprendido entre 1 y 10 (línea 140). A continuación tu ordenador imprimirá el elemento R-ésimo de la lista (línea 160). Una vez que ha sido seleccionado un elemento, hay que eliminarlo del banco de datos para que no resulte elegido por segunda vez. La línea 170 se ocupa de esto sustituyendo el nombre del elemento por una cadena vacía. Al seleccionar un nuevo elemento y antes de imprimirlo, se comprueba si corresponde a una cadena vacía (en la línea 170). Si esto ocurre, se procede a una nueva selección.

Normalmente, el colectivo de datos del que se obtiene la muestra es bastante más extenso que los diez elementos de los que consta este ejemplo. Para muestrear semejantes volúmenes de datos, el programa anterior sería lento e ineficiente. Supongamos por ejemplo que un encuestador desea extraer 200 nombres de un censo electoral de 60000 votantes de una determinada circunscripción electoral; un programa como el anterior necesitaría buscar en toda la lista de electores 200 veces. Para evitar la larga espera que esto llevaría aparejado, sería mejor utilizar un método de un solo paso.

LA BUSQUEDA DE UN SOLO PASO

En el método de un solo paso se leería toda la lista electoral de arriba a abajo una sola vez. Al ir considerando cada nombre, se toma la decisión de si incluir o no a esa persona en la muestra. Este método resulta fácil de programar; realiza los siguientes cambios en el último programa y vuelve a ejecutarlo:

```
30 LOCATE 9,0:PRINT"MUESTREO
   DE UN SOLO PASO"
50 LOCATE 9,1:PRINT"--------
   -- -- ---- ----"
120 A=N:C=10
130 FOR J=1 TO 10
140 IF A=0 THEN GOTO 190
150 IF RND(1)<=A/C THEN
    PRINT A$(J):GOTO 170
160 C=C-1:GOTO 180
170 A=A-1:C=C-1
```

Si ahora tecleas un 3, para seleccionar tres elementos de la lista, el primero de ellos (Bonn) es el primero que se considera. Si la función RND (línea 150) es menor que 3/10, será seleccionado Bonn. Copenague es el elemento que se considera a continuación. Si Bonn ya está en la selección, Copenague sólo podrá ser seleccionado si la función RND genera un valor menor que 2/9, es decir, la probabilidad de elegir a Copenague es menor si ya se ha elegido a Bonn. Por otra parte, si todavía no ha sido seleccionado Bonn, las probabilidades de Copenague se elevarán hasta 3/9. Las líneas 160 y 170 actualizan las probabilidades. Cuando compares los resultados de las selecciones de este programa con las del anterior, advertirás que el método de un solo paso da las muestras en orden alfabético. Además es un método mucho más rápido, aunque en este caso, al ser pocos los elementos entre los que se puede elegir, apenas se note diferencia en velocidad.

A primera vista podría parecer que con una lista de sólo diez ciudades, el número de posibles muestras es pequeño. Esto no es cierto. De hecho hay 120 posibles muestras diferentes de tamaño 3 y 252 muestras de tamaño 5. En un próximo artículo veremos de qué forma pueden desarrollarse estas ideas para ser utilizadas en el tipo de simulación conocida como modelización.

GANADORES DE LOS MEJORES DE INPUT MSX

En el sorteo correspondiente al número 4 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
David Caballero López	Granada	Jet fighter
G. José Sánchez Martín	Las Palmas	Turmoil
M. Angel Martínez Cotano	Alcalá de Guadaira (Sevilla)	Yie ar kung fu 2
Antonio López Fernández	Lleida	Night Shade
Jaime Soler Castanyer	Terrasa (Barcelona)	Ghostbusters
A. Alfonso Benítez González	Alcalá del Río (Sevilla)	Yie ar kung fu 2
Antonio Guillano Montesinos	Valencia	Profanation
Javier Rosendo López	Cabezón de la Sal (Cantabria)	Profanation
F. José S. Cruz Gil	Mérida (Badajoz)	Knight Lore
J. Manuel Gordillo	Valencia	Gunfright

MSX-DOS: POTENTE Y VERSATIL

- QUE ES MSX-DOS
- TRABAJANDO CON DISKETTES
- TIPOS DE COMANDOS
- LA POTENCIA DE LOS COMANDOS INTERNOS

El sistema operativo para manejo de discos MSX-DOS**, hermano menor del conocido** MS-DOS **de los ordenadores de 16 bits, es una formidable herramienta para trabajar con *diskettes*. Te contamos cómo es y cómo se maneja.**

Dentro de la jerga de los ordenadores, muchas veces una auténtica «sopa de letras», se producen algunos malentendidos que conviene aclarar desde el principio. El título de este artículo es uno de los ejemplos que, para los no iniciados, puede producir una cierta confusión. **MSX-2** y **MSX-DOS** quizá parezcan lo mismo pero representan cosas totalmente diferentes. **MSX-2** o **MSX2** se refiere a la segunda generación de ordenadores domésticos del estándar **MSX**, mientras que bajo las siglas **MSX-DOS** se esconde un Sistema Operativo para manejo de ficheros en disco (**DOS** = *Disk Operating System*) diseñado por la casa **Microsoft** para uso exclusivo en los ordenadores **MSX**, si bien con la misma estructura del conocido **MS-DOS** del **IBM PC** o sus compatibles.

A diferencia del Sistema Operativo (**OS** = *Operating System*), que reside permanente en ROM, el **DOS** es un programa en código máquina que es necesario cargar de disco. Sólo puede utilizarse en micros con un mínimo de 64 Kbytes de memoria RAM que dispongan al menos de una unidad de *diskette*.

El Sistema Operativo de un microordenador es un programa básico en código máquina escrito para ese micro en concreto y sin el cual no puede funcionar. Es el encargado de coordinar el entendimiento entre el usuario, el *hardware* (partes interiores y periféricos) y el resto del *software* (como pueden ser programas en los diversos lenguajes).

El **DOS** es por tanto un complemento al **OS**, ampliando sus posibilidades para el eficaz menejo de programas, ficheros de datos, etc. en discos y periféricos como la pantalla e impresora, tal como iremos viendo en este artículo. El **DOS** debe estar escrito, por supuesto, para un **OS** en particular. Igualmente los lenguajes de programación que se utilicen (BASIC interpretado, BASIC compilado, Fortran, Pascal, etc) deben estar preparados para «rodar» bajo el sistema operativo **MSX-DOS**. La casa **Microsoft** garantiza que todos sus lenguajes de programación podrán trabajar en **MSX**.

Hay que aclarar que el **MSX DISK BASIC** es una variante o complemento del BASIC de **Microsoft** que incorpora (estándar en **MSX2** y opcional con cartucho en **MSX**) ciertas facilidades para el manejo de discos y periféricos, pero no es compatible con el **MSX-DOS**. El nombre de los comandos es diferente (Ej. para ver el contenido de un disco se utiliza FILES en **MSX DISK BASIC** y DIR en **MSX-DOS**). De hecho, ambos sistemas operativos están alojados en páginas distintas de la memoria y no pueden utilizarse simultáneamente. El primero puede utilizarse desde el BASIC mientras que para el segundo hay que «salirse del BASIC» (CALL SYSTEM) para «pasar a DOS» o hacer la maniobra contraria mediante el comando «BASIC». Lo que si es común en ambos casos es el formateo del disco y la estructura de almacenamiento y control de ficheros.

A partir de aquí nos centraremos en presentar de una forma global las posibilidades y peculiaridades del **MSX-DOS**.

El disco que contiene el sistema operativo **MSX-DOS** contiene dos ficheros básicos: MSXDOS.SYS y COMMAND.COM. El primero de ellos contiene el programa del Sistema Operativo de Disco propiamente dicho mientras el segundo es un programa que sirve para procesar los diversos comandos que se desea ejecutar, poniendo a su vez en marcha los programas adecuados. A este programa se le suele denominar «procesador de comandos».



Una vez cargado en memoria el primero de los ficheros (los detalles los veremos al tratar del AUTOEXEC.BAT) el sistema busca en el disco el fichero COMMAND.COM, lo carga en memoria y lo ejecuta, tras lo cual aparece en pantalla algo parecido a lo siguiente:

```
MSX-DOS Version 1.00
Copyright 1984 by Microsoft
COMMAND Version 1.00
Current date is SUN 1-01-1984
Enter new date:
Currente time es 08:16:22.15p
Enter new time:
```

Los números de las versiones serán los que correspondan al *software* que estemos utilizando y los textos aparecen en inglés (al menos no conocemos ninguna versión española).

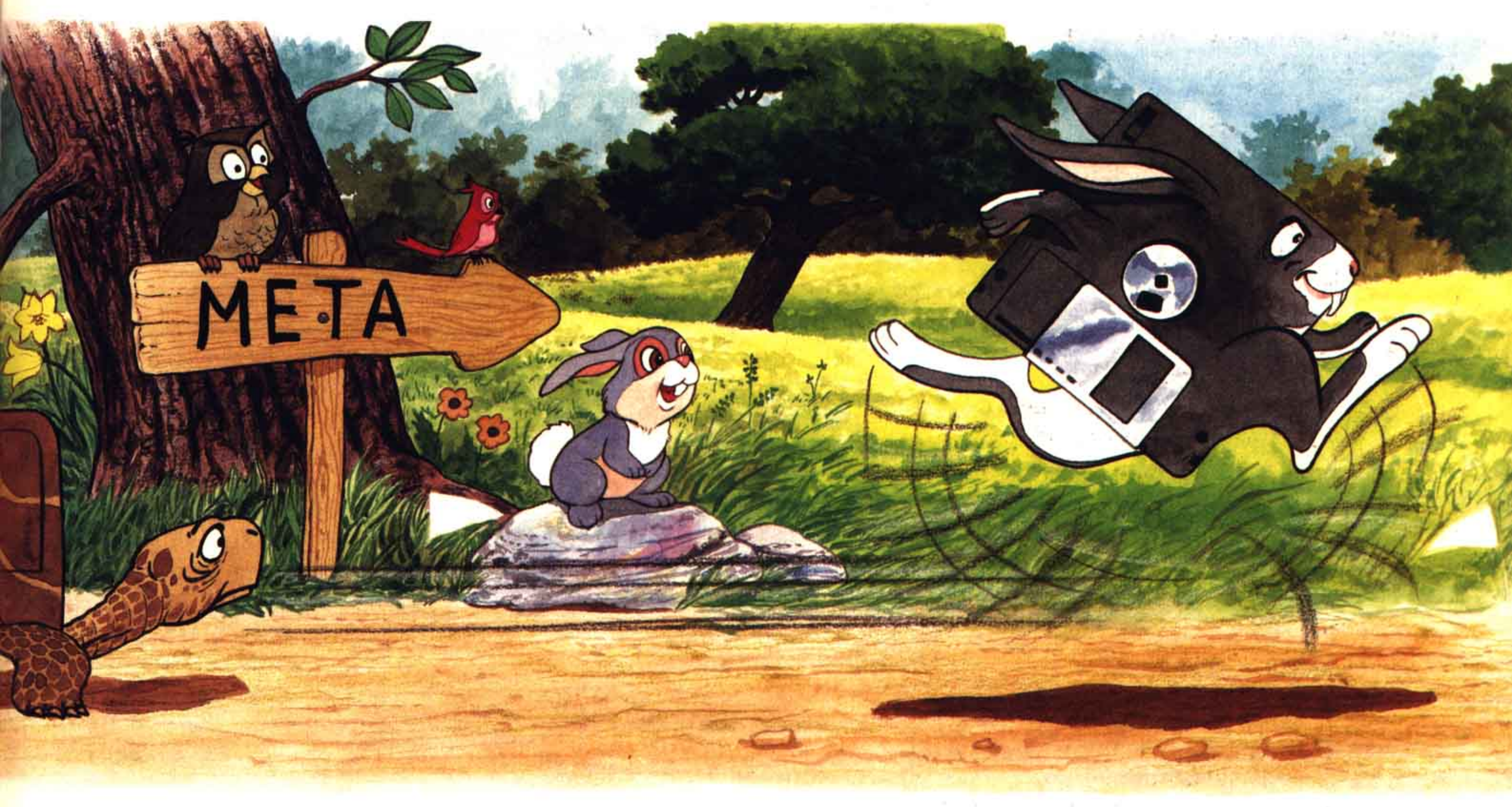
El ordenador nos presenta el día de la semana y la fecha que tiene como actual (Ej. currente date is SUN 1-01-1984) o el valor que tiene por defecto. Si el ordenador dispone de reloj calendario alimentado por batería conservará valores reales aún cuando se desenchufe el equipo, como es el caso de **MSX2**.

basta pulsar RETURN para ignorar la pregunta. Los nuevos valores que deseemos introducir debemos separarlos por «:». Si la hora es superior a las 12 el ordenador resta 12 y situa «p» al final para indicar que es por la tarde (post meridiam).

Una vez introducidos estos parámetros básicos el ordenador nos contestará normalente con «A>» para indicarnos que está listo para trabajar en DOS («>») desde la unidad «A», a menos que los hayamos cargado desde otra unidad, por ejemplo la C, lo

tenido del disco situado en la unidad B, aún cuando tuviésemos metidos más discos en las unidades A o C, por ejemplo.

Conviene añadir que en el caso de utilizar una sola unidad, situación bastante normal, el concepto «*drive* A» y «*drive* B» puede resultar confuso. Realmente el ordenador dispone de una sola unidad y le da lo mismo unidad A que unidad B pero a nosotros nos puede resultar de gran ayuda en el aspecto operativo. En este caso debemos interpretar «coloca/retira disquete A» o «coloca/retira disquete B» de la única unidad (boca) que existe. La explicación de una operación determinada puede darse, por lo tanto, de forma análoga tanto si tenemos dos unidades de disquete como si disponemos de solo una y necesitamos alternar discos.



Si deseamos mantener los valores basta con RETURN. En caso contrario deberemos indicarlos (día, mes, año) mediante los separadores «-» o «/». Los valores de día y mes deben mantenrse entre 1-31 y 1-12 respectivamente ya que en caso contrario el ordenador nos volverá a solicitar los datos. Respecto al año podemos fijar cuatro cifras o si sólo deseamos utilizar las dos últimas el ordenador supondrá que debe añadir 1900 si los valores dados se encuentran entre 80-99 ó 2000 si lo están entre 00-79.

Si el ordenador nos solícita la hora pero no disponemos de reloj interno cual no es lo usual, en cuyo caso aparecería «C>». Si no se especifica otra cosa la unidad por defecto es la «A», pero si se desea trabajar con otra unidad por defecto basta con pulsar la letra identificativa de la unidad en cuestión seguida de «:» RETURN. Veamos un ejemplo:

```
A> (Situacion actual)
A>B: (Eleccion de B como
drive por defecto)
B> (nueva situacion)
```

Si en este momento tecleasemos DIR nos aparecería en pantalla el con-

FORMATEADO

Antes de poder utlizar un disco para manejar ficheros bajo **MSX-DOS** debemos «formatearlo» convenientemente mediante el comando FOR-

MAT. Durante esta operación se hace una asignación e identificación de la superficie magnética del disco de forma que el ordenador pueda dirigirse posteriormente con precisión a cualquier zona de la misma para grabar o leer información. Se generan, además, dos áreas donde el sistema irá situando en cada momento los nombres, longitud y situación de cada fichero así como la distribución del espacio disponible. Estas áreas son el Directorio y *File Allocation Tables* (tabla de asignación de ficheros).

El proceso de formateo es independiente de que el disco haya sido usado antes o no (si existe alguna información previa quedará destruida). Si existe alguna zona defectuosa quedará automáticamente descartada para uso posterior y si el volumen a descartar queda por encima de unos límites preestablecidos el programa nos sugerirá que probemos con otro disco.

Como respuesta al comando FORMAT el ordenador solicitará el nombre de la unidad donde deseamos realizar la operación:

```
Drive name ? (A, B, C, D)
```

Si pulsamos RETURN optaremos por la unidad elegida por defecto.

Tan pronto se haya concluido la operación aparecerá el mensaje: «*Format complete*».

En cualquier caso debe seguirse las instrucciones suministradas con cada unidad de disco ya que pueden requerirse algunos datos más como en el caso de utilizar el formateo a dos caras.

IDENTIFICACION DE FICHEROS

Todos los ficheros de un disco deben estar identificados inequívocamente, o lo que es lo mismo no puede haber dos ficheros con el mismo nombre. Un fichero queda identificado de la siguiente forma:

```
[<unidad de disco:>]
<nombre de fichero>
[<extension>]
```

Por ejemplo:

```
B:CLIENTES.BAS
```

Es un fichero con programa en BASIC denominado CLIENTES y situado en la unidad B.

Todo lo situado entre [] es opcional.

Como nombre de fichero puede elegirse cualquier combinación de 1 a 8 caracteres (las letras puede ser mayúsculas o minúsculas). En el caso de la extensión el número de caracteres oscila entre 1 y 3 precedidos por un punto (si no se indica extensión alguna no debe colocarse el punto).

La «extensión» es algo que complementa al nombre y puede ayudar a clasificar los ficheros de una forma más operativa. Por ejemplo, podemos elegir la extensión <.TXT> para todos aquellos ficheros donde sólo tengamos texto o <. BAS> para aquellos con programas BASIC, etc.

Hay ciertos nombres reservados para funciones específicas que no pueden utilizarse.

COMODINES (*wild cards*)

Al igual que en la baraja existen cartas denominadas comodines que pueden tomar el lugar de cualquiera de ellas, a la hora de manejar nombres de ficheros también podemos hacer uso de ciertos símbolos comodines (*wild cards*) lo cual nos ofrece ciertas ventajas operativas. En **MSX-DOS** se utilizan dos tipos: «?» y «*».

El símbolo «?» puede sustituir a uno de los símbolos autorizados en el nombre o la extensión de un fichero, tantas veces como se desee. Por ejemplo, a la hora de localizar el fichero CL??NT?.B?N, el ordenador puede presentarnos CLIENTE. BCN, CLIENTA.BHN, CL70NTP.B0N o cualquier otro de los que existan en el Directorio que encaje con las cuatro variables prefijadas.

El comodín «*» es más potente ya que «*» puede sustituir a todo un nombre o lo que resta de él. Por ejemplo «*.BAT» se refiere a todos los ficheros de tipo (extensión) BAT, sea cual sea su nombre. CLI*DAT se refiere a todos aquellos ficheros tales como CLIENERO.DAT, CLIFEB.DAT, CLIDEUD.DAT, etc. A la hora de eliminar de nuestro disco todos los ficheros <.DAT> podemos hacerlo uno a uno o simplemente con «DEL *.DAT». Ya podemos imaginar lo que puede ocurrir si no tomamos precauciones..., y más aún si el comando es «DEL *.*» (el ordenador, ante este comando nos pide confirmación a fin de que podamos rectificar a tiempo). Debemos utilizar la instrucción «DEL *.*» en lugar de FORMAT si lo que deseamos es simplemente «limpiar» un disco para su reutilización.

TIPOS DE COMANDOS

En **MSX-DOS** existen los denominados comandos «internos» y comandos «externos». Los primeros son los nuevos comandos que se incorporan al sistema al cargar el **DOS** y se ejecutan de inmediato. Son los siguientes:

```
BASIC
COPY
DATE
DEL (ERASE)
DIR
FORMAT
MODE
PAUSE
REM
REN (RENAME)
TIME
TYPE
VERIFY
```

Los comandos «externos» son realmente ficheros o programas del tipo <.COM> o <.BAT> que residen en disco y lógicamente han de ser cargados de éste antes de ejecutarse. Por tanto, cada vez que se desee utilizar uno de estos comandos debe estar operativo el disco que los contenga, lo cual ha de tenerse en cuenta, sobre todo cuando se opere con una sola unidad de disco. Los comandos externos se llaman o invocan sólo por su nombre ignorando la extensión.

Lo más interesante de los comandos

externos es que podemos definirlos de acuerdo a nuestras necesidades.

Vamos a comentar los aspectos más importantes de cada uno de los «comandos interiores».

BASIC [<espec. de fichero>]

Si se utiliza simplemente BASIC se abandona el **DOS** para pasar al *Disk-Basic*. Para hacer la maniobra opuesta se utiliza «CALL SYSTEM». Cuando se indican datos de algún fichero en especial se supone que éste es un programa que debe cargarse y ejecutarse automáticamente una vez situados en el BASIC.

COPY <espec. de fichero orig.> [<espec.> de fichero dest.>]

Copia uno o más ficheros de un disco a otro manteniendo o cambiando los nombres. Incluso pueden hacerse copias dentro del mismo disco, aunque en este caso resulta imprescindible cambiar el nombre.

Si no se especifica unidad en la segunda opción se considerará se trata de la unidad elegida por defecto. Si se indica algún nombre se dará éste a la copia y en caso contrario se mantendrá el mismo.

Ej. COPY C:GASTOS.PAR B:PAGOS.ROM

Copiará el fichero GASTOS.PAR situado en la unidad C en el fichero PAGOS.ROM pero en la unidad B.

Durante el proceso de copia se puede aprovechar para «concatenar» o «fundir» ficheros (lógicamente deben ser del mismo tipo) separando por «+» las especificaciones de ficheros correspondientes.

Ej. COPY C:PEPE.AB + LUIS + D:JUAN NOMBRE

Fundirá el fichero PEPE.AB de la unidad C con el LUIS de la unidad elegida por defecto (quizá A) y el JUAN de la unidad D en un nuevo fichero denominado NOMBRE situado en la unidad por defecto.

Se puede ir incluso más lejos y dar la orden de copiar , por ejemplo, todos los ficheros que tienen un determinado tipo de extensión dentro de un fichero de forma que queden refundidos en la misma unidad con otro nombre. Ej. COPY *SDF SUMA.TXT.

Las combinaciones pueden ser muy grandes.

Las concatenaciones se realizan normalmente con ficheros ASCII, pero también pueden hacerse con ficheros binarios. El primer caso es el contemplado por defecto.

DATE [<mm-dd-aa>]

Al principio de este artículo ya hicimos algunas precisiones sobre la forma de manejar el calendario.

Con DATE conocemos la fecha que tiene el sistema y mediante los parámetros opcionales podemos variarla.

Si no se incluye DATE en el AUTOEXEC.BAT (comando de arranque inicial) no aparecerá automáticamente la fecha en pantalla.

Cada vez que se registre o modifique un fichero quedará reflejada la fecha que tiene el sistema en ese momento.

DEL <espc. de fichero>

Sirve para borrar o cancelar un fichero en concreto o un conjunto de ellos si se hace uso de símbolos comodines, tal como ya se ha visto.

DIR [espec. de fichero] [/P] [/W]

Se trata del comando para conocer el contenido del directorio de un disco o el contenido selectivo que nos interese.

DIR presenta en pantalla el contenido de un disco indicando al lado de cada nombre de fichero su longitud en bytes, así como la fecha y hora de su creación o última actualización.

DIR *.TPT haría la misma presentación pero referido sólo a los ficheros cuya extensión es «.TPT» referido a la unidad por defecto.

Cuando los ficheros son muy numerosos y no caben todos en una pantalla, con sus datos de detalle respectivos, tenemos la opción de utilizar el interruptor «/P» y que aparezcan por pantallas completas, en lugar de producirse un *scrolling*, y podamos solicitar manualmente la siguiente pantalla, y así sucesivamente. Si en su lugar utilizamos «/W» los datos se compactarán quedando sólo los nombres, uno a continuación de otro sin las informaciones complementarias.

FORMAT

Su uso ya se vió en la primera parte del artículo.

MODE <anchura de pantalla>

Establece una anchura de pantalla prefijada en número de caracteres.

PAUSE [comentario]

Suspende la ejecución de un fichero *batch* para darnos la oportunidad de realizar alguna maniobra. La operación se reanuda pulsando cualquier tecla (excepto CTRL-C).

Al producirse la parada aparece en pantalla el mensaje «Strike a key when ready...» (Pulse una tecla cuando este listo...). Si se desea, antes de este mensaje podemos introducir el nuestro, si lo hemos escrito a continuación de PAUSE.

Si pulsamos CTRL-C el ordenador preguntará si queremos seguir o abortar el proceso en curso y retornar al nivel de comando del sistema operativo. Resulta muy útil fraccionar los procesos en módulos.

REM [comentario]

Permite hacer observaciones en los programas *batch* a efectos de documentación o clarificación.

REN <espec. de fichero> <nuevo nombre> de fichero>

Sirve para cambiar el nombre a un fichero determinado situado en cualquier unidad. Puede hacerse también una sustitución en cadena mediante comodines, lo cual resulta muy útil si, por ejemplo se desea cambiar el tipo de extensión a un conjunto ficheros. REN *.PNT *.TTY cambiará el nombre de todos los ficheros con extensión PNT en ficheros con el mismo nombre pero extensión TTY, en la unidad por defecto, puesto que no se ha especificado otro.

TIME [<hh>[:mm>[:ss>]]]

Permite conocer la hora del reloj interno (no todos los ordenadores **MSX** disponen de él) o modificarla mediante los oportunos parámetros.

TYPE <espec. de fichero>

Permite visualizar el contenido del fichero que se especifique. Dependiendo del tipo de fichero, la presentación en pantalla puede resultar totalmente ininteligible a primera vista.

VERIFY (ON : OFF)

En situación de VERIFY ON, cada vez que se escribe algo en disco se efectúa una verificación de que la escritura ha sido correcta, lo cual añade una gran fiabilidad a costa de una mayor lentitud y trabajo del disco. VERIFY OFF desactiva esta situación.

PROCESO BATCH

En lugar de teclear y ejecutar sucesivamente una serie de comandos se pueden incluir en un fichero de tipo *batch* (<.BAT>) y ejecutarlo y conservarlo en disco para cuando deseemos ejecutarlo.

Uno de los ficheros más importantes de este tipo es el denominado AU-



TOEXEC.BAT. Cuando se pone en marcha el **MSX-DOS** busca siempre la existencia de este fichero y ejecuta su contenido.

Un AUTOEXEC.BAT se crea de la manera siguiente:

```
COPY CON:AUTOEXEC.BAT
...........
comandos internos o externos
...........
CONTROL-Z
```

Siempre que se desee ejecutar un fichero .BAT debe invocársele por su nombre, sin extensión.

Con un fichero AUTOEXEC.BAT podemos hacer, por ejemplo que nada más arrancar el sistema nos aparezca un determinado dibujo en pantalla, pasados unos segundos aparezca un menú y a continuación el sistema nos permita ejercer ciertas acciones, quizá dependiendo del día de la semana.

Quizá una misma estructura de fichero .BAT pueda valernos para un gran número de programas pero con distintos valores de parámetros. El **MSX-DOS** nos permite utilizar hasta diez parámetros variables (%0 - %9), los cuales se valoran según convenga en cada caso a la hora de llamar al fichero para ejecutarlo. Un ejemplo de fichero *batch* puede ser:

```
A> COPY CON:DEMOS.BAT
TYPE %0.BAT
COPY %2.TXT %1.NTF
BASIC RELOJ
CONTROL-Z
(%0 %1 %2)
```

Si invocamos DEMOS GASTO COBRO lo que estaremos pidiendo es ejecutar el fichero DEMOS, viendo primeramente su contenido, luego copiando el fichero COBRO.TXT en el fichero GASTOS.NTF y finalmente pasar a BASIC para ejecutar el programa RELOJ.



AYUDAS PARA LA EDICION

Aún cuando el proceso de edición puede realizarse sin ayudas especiales, el **MSX-DOS** ofrece ciertas facilidades, mediante el uso de ciertas teclas, gracias a las cuales se puede obtener cierta comodidad y seguridad cuando se trata de manejar muchas instrucciones o éstas son muy parecidas.

Cuando se ha tecleado una instrucción en la línea de comando y se está de acuerdo con su contenido se finaliza la operación con RETURN con lo cual se consigue que el sistema acepte como propia la instrucción y además la copie en un *buffer* especial para su posible uso posterior.

Si en lugar de teclear una nueva instrucción queremos aprovechar una parte de la anterior, ahorrando trabajo y errores de transcripción, podemos hacer uso de 9 conjuntos de teclas (consultar el manual del **DOS** para mayor detalle) con las cuales podemos copiar todo, copiar hasta un lugar determinado, copiar desde un lugar, insertar o eliminar caracteres, etc.

ERRORES EN EL MANEJO DE DISCOS

Si durante la ejecución de un comando o un programa se produce un error, el **MSX-DOS** repite la operación hasta tres veces, y si después de eso no puede completarse la ejecución, el sistema presenta en pantalla el siguiente mensaje, esperando nuestra respuesta:

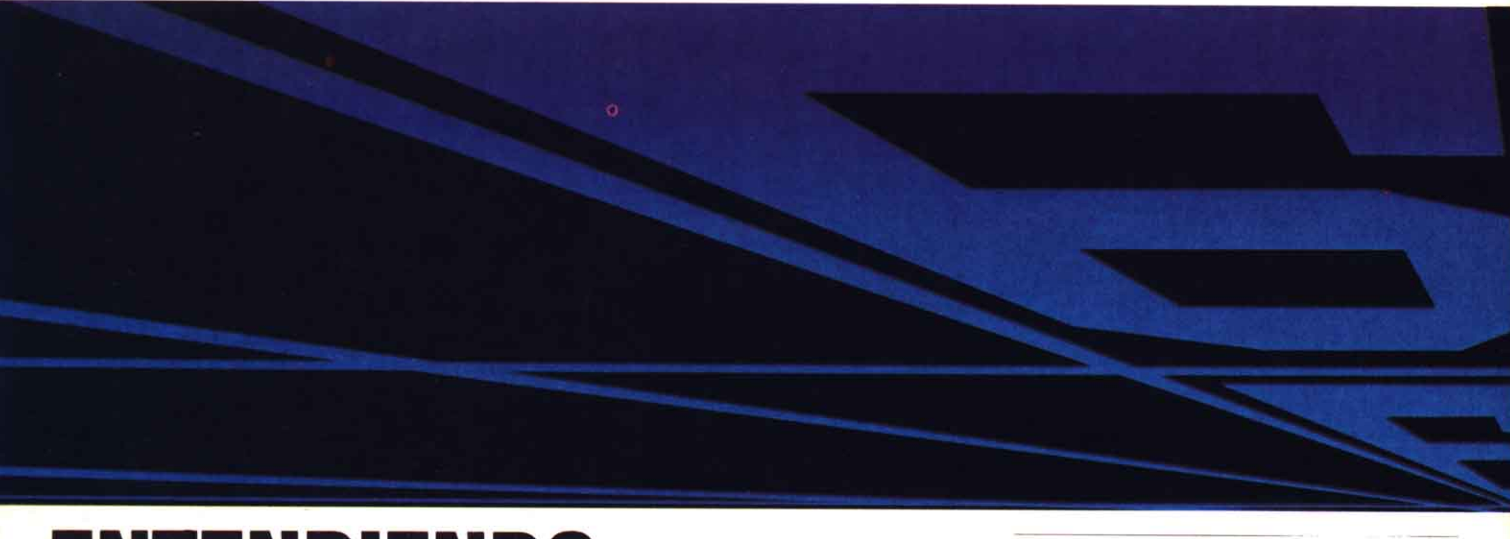
<tipo>error < I/O action > unidad <x>

El tipo puede ser: «*Write protect*» (protegido contra escritura), «*Not ready*» (No preparado) ó «*Disk*».

La acción I/O puede ser: «*reading*» (leyendo) o «*writing*» (escribiendo).

Nuestra respuesta puede ser: A (*Abort*) si queremos abortar la operación, I (*Ignore*) si deseamos pasar por alto la parte donde se produce el error y continuar o finalmente R (*Retry*) si pretendemos repetir la operación (dando por supuesto que hemos subsanado la causa del error).





ENTENDIENDO LOS CODIGOS ASCII

- ¿QUE ES EL CODIGO ASCII?
- USO DE NUMEROS EN
- LUGAR DE CARACTERES
- PROGRAMA PARA IMPRIMIR
- MENSAJES CIFRADOS

El código ASCII empleado por los ordenadores tiene las suficientes semejanzas para permitirles literalmente «hablar» unos con otros. Este «lenguaje común» tiene también otros usos...

Cada tecla o combinación de teclas de tu ordenador está representada por un código electrónico único. Estos códigos se representan en BASIC por una serie de valores decimales que van desde 0 hasta 255. Por ejemplo, a la letra A le corresponde el valor decimal 65, a la B el 66, etcétera. Cuando tecleas letras o palabras cualesquiera, lo que el ordenador realmente almacena son estos códigos.

Los valores correspondientes a cada configuración de teclas no son los mismos en todos los ordenadores, pero afortunadamente existe al menos una cierta estandarización en la forma en que dichos valores se describen y utilizan.

Este conjunto de valores se designa por las siglas ASCII que significa **Código Americano Normalizado para Intercambio de Información** (*American Standard Code for Information Interchange*). Este código se elaboró para disponer de un medio por el que los ordenadores pudieran transferirse datos de unos a otros. Los **Commodore**, **Spectrum** y **MSX** utilizan este código, al menos en parte. El **ZX81** de **Sinclair**, en cambio, tiene un código completamente diferente, único para esta máquina.

EL CODIGO ASCII

El conjunto completo de caracteres ASCII no es igual para todos los ordenadores. Pero hay bastantes semejanzas. El mayor paralelismo se produce en el intervalo desde 33 hasta 90, que cubre todos los símbolos normales: signos de puntuación, números y letras mayúsculas. Hay una forma muy sencilla de encontrar el código ASCII de un carácter cualquiera; para ello no tienes más que teclear PRINT ASC(«X») y te aparecerá el código ASCII de «X» o de cualquier otro carácter en el que estés interesado. El siguiente programa te permite introducir los caracteres con más facilidad.





```
10 SCREEN0:CLS
15 PRINT"Teclea cualquier
   letra,":PRINT"numero o
   caracter"
20 INPUT A$
30 PRINT"El codigo ASCII de
   ";A$;" es ";ASC(A$)
40 GOTO 20
```

La función inversa de ASC es CHR$, que realiza la conversión del número de código a su correspondiente caracter. El programa que viene a continuación convierte en caracteres todos los códigos ASCII comprendidos entre el 33 y el 90:

```
10 SCREEN0:CLS
20 PRINT"CODIGO ASCII",
   "CARACTER"
30 FOR J=33 TO 90
40 PRINT J,CHR$(J)
50 FOR D=1 TO 100
   :NEXT D
60 NEXT J
```

Las letras minúsculas del alfabeto se extienden en el margen de 97 a 122, como puedes comprobar cambiando por 122 la última cifra que figura en la línea 20. Si ahora cambias de nuevo dicha cifra por 255, obtendrás la representación del conjunto completo de caracteres. En el manual de tu equipo puedes ver el listado completo de los mismos.

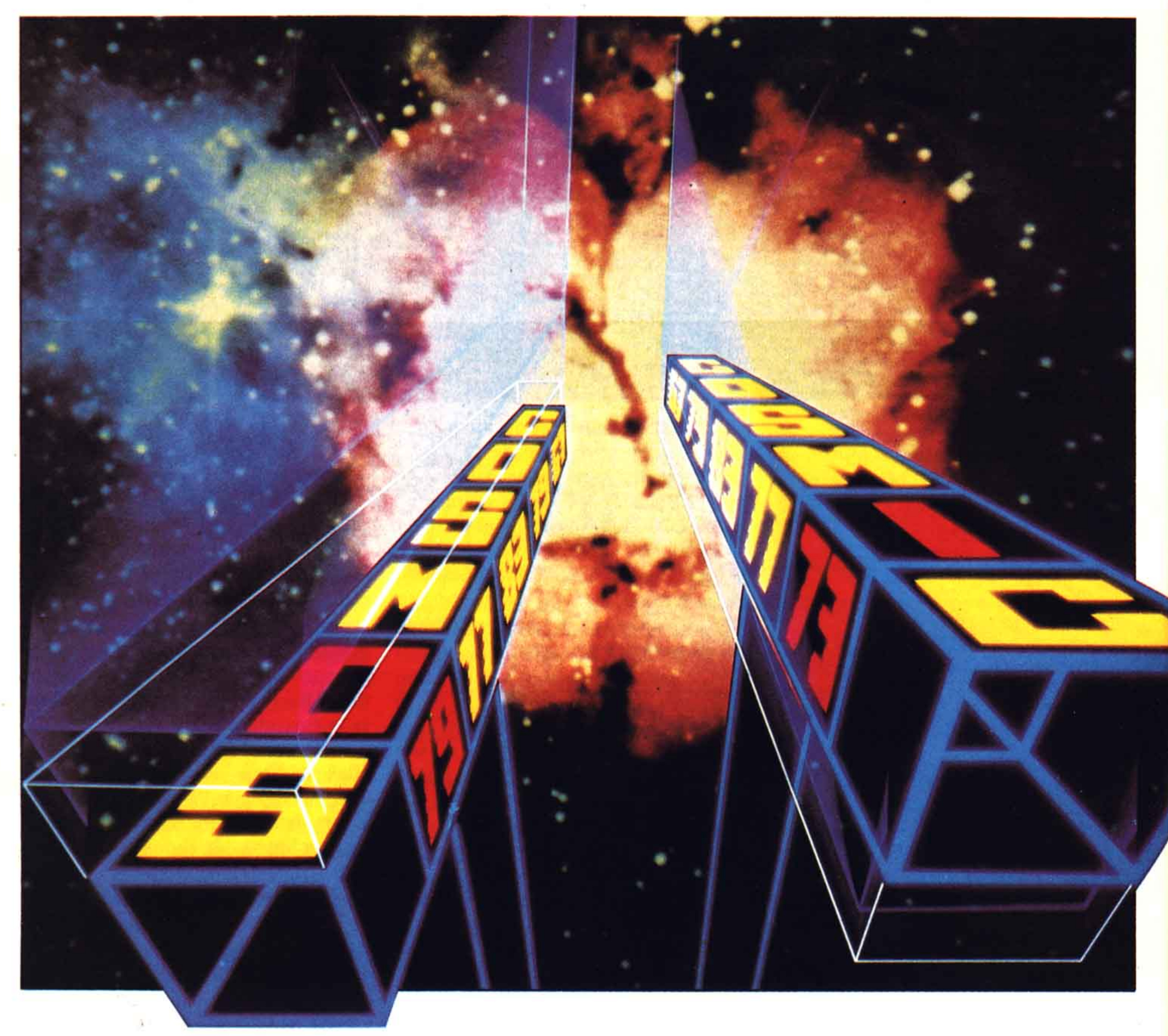
USO DEL CODIGO ASCII

La ventaja de utilizar un número en vez de un carácter, consiste en que puedes modificar matemáticamente dicho número de varias maneras. Si después de hacerlo imprimes el carácter del nuevo número, obtendrás una letra diferente. Esta es la base de muchos programas de escritura de códigos. Se trata de una utilidad de gran valor, ya que no es posible manipular directamente una letra aplicándole una operación matemática.

Los códigos más sencillos se limitan a añadir una cantidad constante a cada número, con lo que resulta de hecho una traslación de la letra en el alfabeto. Por ejemplo, la A se convierte en G, la B en H y así sucesivamente. Pero estos códigos resultan muy fáciles de romper; después de todo, no sería muy difícil escribir un programa de ordenador para probar cada una de las 26 combinaciones posibles, siendo muy fácil detectar de una ojeada cuál es la correcta.

Para que el programa tenga alguna eficacia, tiene que hacer cosas más complicadas con los números; el programa siguiente utiliza una palabra código de forma que cada letra del mensaje se altera de una manera diferente. Este tipo de codificación es muy difícil de romper, a menos que se sepa cuál es la palabra código.

```
10 SCREEN0:CLS:PRINT"ESCRIBE
   EN MAYUSCULAS":PRINT
   :PRINT
```



```
30 INPUT"Cual es la PALABRA
   CLAVE";C$
40 CLS:PRINT"Escribe tu
   mensaje:-"
50 INPUT MS$
60 CP=CP+1:IF CP>LEN(C$)
   THEN CP=1
70 PM=PM+1
80 IF PM>LEN(MS$) THEN 200
90 F$=MID$(MS$,PM,1)
100 IF F$<"A" OR F$>"Z"
    THEN GOTO 150
110 F=ASC(F$)+ASC(MID$
    (C$,CP,1))-65
120 IF F>90 THEN F=F-26
130 CD$=CD$+CHR$(F)
140 GOTO 60
150 IF F$<"0" OR F$>"9"
    THEN CD$=CD$+F$
    :GOTO 70
160 F=ASC(F$)+ASC(MID$
    (C$,CP,1))-48
170 IF F>57 THEN F=F-10
    :GOTO 170
180 CD$=CD$+CHR$(F)
190 GOTO 60
200 PRINT:PRINT"El mensaje
    codificado es:-"
210 PRINT:PRINT CD$
220 END
```

El programa te pide que introduzcas la palabra código y el mensaje, y a continuación imprime la versión cifrada. El mensaje ya está protegido y sólo lo podrá descifrar alguien que conozca la palabra clave.

La forma en que trabaja el programa es muy sencilla y se parece mucho al programa de codificación simple, con la salvedad de que en vez de su-

mar una cantidad constante a cada letra, se le suma el código ASCII de la primera letra de la palabra código a la primera letra del mensaje, el de la segunda letra de la palabra código se suma a la segunda letra del mensaje, etcétera. Cuando se llega al final de la palabra código, el ciclo comienza de nuevo.

Si al sumar los números se obtiene un resultado mayor que 90 —lo que supone irse más allá de la letra Z— se restan 26 para mantenerse dentro del margen correspondiente a las letras del alfabeto. Los números reciben un tratamiento separado en las líneas 150 a 180, a fin de mantener sus códigos ASCII entre 48 y 57, que corresponden a los números 0 a 9.

El programa de descifrado es muy parecido al primero; la única diferencia consiste en que se han cambiado unos cuantos +s y −s. La numeración de las líneas es coherente con la del programa anterior, a fin de que puedas almacenar el programa combinado. Tienes además unas cuantas líneas adicionales que te permiten seleccionar la codificación o decodificación de un mensaje.

```
12 INPUT"(C) CODIFICAR O
   (D) DECODIFICAR";A$
14 IF A$="D" THEN GOTO 400
16 IF A$<>"C" THEN GOTO 12
400 CLS
410 INPUT"CUAL ES LA PALABRA
    CLAVE";C$
420 PRINT:PRINT"ESCRIBE EL
    MENSAJE:-"
430 INPUT M$
440 CP=CP+1:IF CP>LEN(C$)
    THEN CP=1
450 PM=PM+1
460 IF PM>LEN(M$) THEN
    GOTO 580
470 F$=MID$(M$,PM,1)
480 IF F$<"A" OR F$>"Z"
    THEN 530
490 F=ASC(F$)-ASC(MID$
    (C$,CP,1))+65
500 IF F<65 THEN F=F+26
510 D$=D$+CHR$(F)
520 GOTO 440
530 IF F$<"0" OR F$>"9"
    THEN D$=D$+F$:GOTO 450
540 F=ASC(F$)+ASC(MID$
    (C$,CP,1))+48
550 IF F<48 THEN F=F+10
    :GOTO 550
560 D$=D$+CHR$(F)
570 GOTO 440
580 PRINT"EL MENSAJE
    DECODIFICADO ES:-"
590 PRINT D$
600 END
```

Digamos de paso que si realmente quieres conseguir que tu código sea muy difícil de romper, siempre puedes volver a cifrar tu mensaje cifrado utilizando una segunda palabra código. Acuérdate de colocarlas en el orden correcto cuando descifres de nuevo el mensaje.

COMPARACIONES

Una de las principales aplicaciones de los valores del código ASCII es la comparación de cadenas de caracteres. Dichas comparaciones las hace el ordenador carácter a carácter y de izquierda a derecha, hasta que se llega al final de la cadena. Supongamos, por ejemplo, que hay que comparar dos cadenas tales como «COSMOS» y «COSMIC». El ordenador primero compara las dos C, después las O, las S y las M. Seguidamente se compara la O de la primera cadena con el correspondiente carácter de la otra cadena, que es la I.

Lo que se compara no es un conjunto arbitrario de valores alfabéticos, sino el valor ASCII de cada carácter. El valor ASCII de la O es mayor que el de la I, por lo que en esta comparación «COSMOS» tiene un valor mayor que «COSMIC». Observa que no se utilizan las sumas de los valores ASCII de las letras para determinar cuál es la cadena que tiene un mayor valor. Cada letra se compara individualmente.

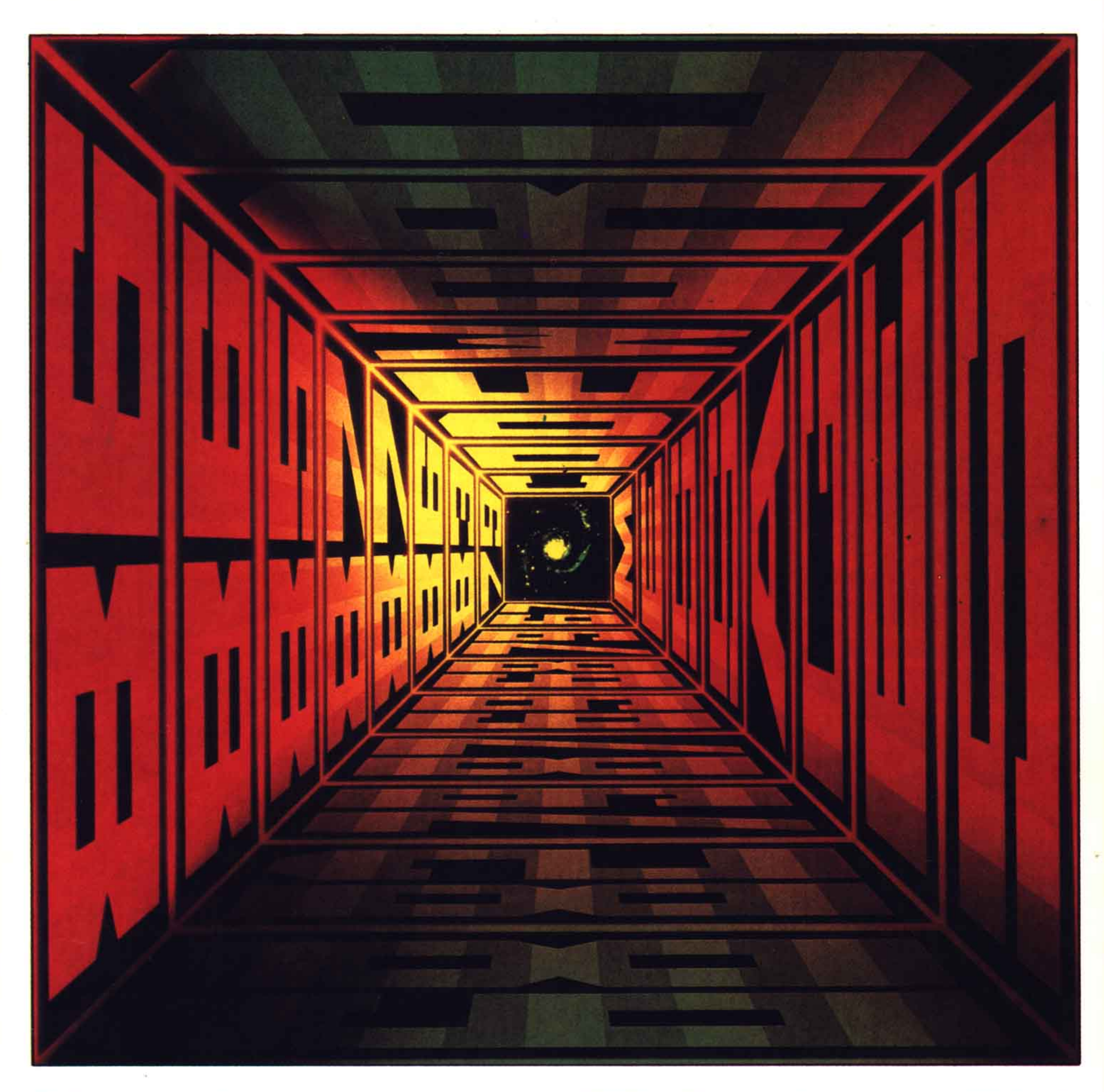
En la tabla 1 tienes como ejemplo algunas comparaciones que muestran posibles valores de A$ y B$, junto con los códigos ASCII de los caracteres que intervienen. Es importante comparar los valores uno a uno, el primer carácter de A$ con el primero de B$, siguiendo así en toda la extensión de la palabra más corta.

Como puedes observar en los ejemplos segundo y tercero, se puede escribir la relación de más de una forma. Fíjate también en el último ejemplo, en el que a igualdad de todo lo demás, se considera que es mayor la cadena de caracteres más larga.

COMPROBACION DE LAS ENTRADAS

La función ASC actúa únicamente sobre el primer carácter de una cadena. Así, la instrucción PRINT ASC(«USA») dará como resultado el número 85, que es el código ASCII de la letra «U». Este hecho resulta muy útil en la comprobación de las entradas que se suministran a los programas por medio del comando INPUT. Por ejemplo, en el anterior programa de escritura de códigos, la línea 12 te pedía que introdujeras la letra C para cifrar o la letra D para descifrar. El caso es que si en lugar de teclear sólo la tecla C o sólo la tecla D, a alguien se le ocurre teclear la palabra complete CIFRAR o DESCIFRAR, el programa no la admitirá. Si quieres evitar este problema y admitir las palabras anteriores o cualquier otra que

A$	ASCII	B$	ASCII	RELACION
ABC	65,66,67	ABC	65,66,67	A$ = B$
ABD	65,66,68	ABCD	65,66,67,68	A$ > B$
ABD	65,66,68	ABCD	65,66,67,68	A$ <> B$
ABC	65,66,67	Abc	65,97,98	A$ < B$
COSMI	67,79,83,77,73	COSMO	67,79,83,77,79	A$ < B$
$1	36,49	$1.0	36,49,46	A$ < B$



empiece por C o por D como respuesta válida, puedes escribir de esta otra forma las líneas 14 y 16:

```
14 IF ASC(A$)=68 THEN
   GOTO 400
16 IF ASC(A$)<>67 THEN
   GOTO 12
```

CODIGOS DE CONTROL

Algunos códigos ASCII no van asociados a ningún carácter. Por ejemplo, cuando pulsas la tecla ENTER o RETURN, o la tecla INS o la tecla HOME, el ordenador almacena el valor ASCII correspondiente a la tecla, pero en lugar de imprimir un carácter en la pantalla, se produce otro tipo de acción. En el primer caso, el cursor se desplaza al principio de la línea siguiente, con INS el editor de pantalla entra en modo inserción y con HOME se produce el desplazamiento del cursor a la esquina superior izquierda de la pantalla.

El código ASCII de HOME es 11, el de INS es 18 y el de RETURN 13. La acción que lleva a cabo el ordenador cuando se pulsa cualquiera de estas teclas, se puede conseguir en algunos casos enviando el código ASCII correspondiente a través de la instrucción CHR$. Por ejemplo prueba a teclear lo que sigue y pulsa RETURN cuando hayas terminado:

```
PRINT "HOLA";CHR$(10);
"ADIOS"
```

Como verás, aparece la palabra

HOLA, y a continuación se desplaza el cursor a la línea siguiente, pero manteniéndose en la columna en la que se encontraba, antes de que aparezca impresa la palabra ADIOS. Hemos utilizado el código ASCII 10, que corresponde al salto de línea. Si quieres que además de saltar de línea, el cursor se sitúe al comienzo de la misma, tendrás que emplear, además del código 10, el código 13, que se suele denominar «retorno de carro» y que se corresponde con la tecla RETURN. Si quieres probarlo teclea:

```
PRINT "HOLA";CHR$(10);
CHR$(13);"ADIOS"
```

En este caso verás como la palabra ADIOS se imprime exáctamente debajo de la palabra HOLA.

Aquí tienes la manera de encontrar los códigos ASCII de las teclas que no corresponden a caracteres:

```
10 A$=INKEY$:IF A$=""
   THEN 10
20 PRINT ASC(A$)
30 GOTO 10
```

El BASIC de algunos ordenadores incluye códigos o instrucciones especiales para los movimientos de cursor, desde dentro de un programa. En **MSX** puedes hacer lo mismo utilizando los códigos ASCII 28, 29, 30 y 31. Por ejemplo, el código 29 representa el movimiento del cursor una posición a la izquierda, mientras que el código 31 representa un desplazamiento hacia abajo. Prueba a utilizarlos con este programa:

```
10 CLS:FOR J=1 TO 10:PRINT
   "A";CHR$(31);
20 NEXT
30 FOR J=1 TO 10:PRINT "A"
   ;CHR$(31);CHR$(29);
   CHR$(29);
35 NEXT
```

Como verás, se pueden conseguir efectos interesantes sin necesidad de recurrir a un montón de instrucciones TAB o LOCATE.

Hay dos teclas en tu teclado, ESC y SELECT, que aparentemente no sirven para nada. Sin embargo, cada vez que pulsas ESC estás enviándole al ordenador el código 27, y si pulsas SELECT, el código 24. lo que ocurre es que como estos códigos no representan ninguna acción concreta, tú no notas nada. Pero el ordenador sí nota algo. Recibe la señal de que se ha pulsado una de estas dos teclas. Ambas están pensadas para que el programador las utilice donde y cuando quiera. Prueba lo que sigue:

```
10 CLS
20 A$=INKEY$:IF A$=""
   THEN 20
30 IF ASC(A$)=24 THEN
   C=(C+1)MOD16
40 COLOR 15,C
50 GOTO 20
```

Verás que al pulsar la tecla SELECT va cambiando el color de la pantalla. Unos ordenadores utilizan estos códigos de control más que otros. Hay muchos números disponibles: desde el 1 al 31 y unos cuantos por encima del 90. Inicialmente fueron definidos para conectar los ordenadores a impresoras, por lo que los códigos estaban relacionados fundamentalmente con el control de aquellas vetustas máquinas. Naturalmente, en la actualidad los ordenadores están diseñados para ser utilizados con un monitor de vídeo, por lo que la mayoría de los códigos han sido reconvertidos, aunque algunos de ellos se sigan utilizando con las modernas impresoras.

En **MSX**, por ejemplo, los códigos de 0 a 31 se utilizan, además de para las funciones que hemos comentado de limpiar la pantalla, retorno de carro, etc, para imprimir un conjunto de caracteres gráficos. Para ello hay que utilizar dos códigos. El primero es el código 1 que indica que vamos a imprimir un gráfico. El segundo se obtiene sumando 64 al código del gráfico propiamente dicho. Con el programa que sigue podrás hechar un vistazo a todos estos caracteres gráficos, que sin duda te serán de utilidad en muchos de tus programas.

```
10 FOR J=0 TO 31
20 PRINT CHR$(1)+
   CHR$(J+64)
30 NEXT
```

TABLA DE CODIGOS DE CONTROL

DECIMAL	HEX	FUNCION	DECIMAL	HEX	FUNCION
0	0	-	16	10	-
1	1	-	17	11	-
2	2	-	18	12	INS
3	3	-	19	13	-
4	4	-	20	14	-
5	5	-	21	15	-
6	6	-	22	16	-
7	7	BEEP	23	17	-
8	8	BACK SPACE	24	18	SELECT
9	9	TAB	25	19	-
10	A	SALTO LINEA	26	1A	-
11	B	HOME	27	1B	ESC
12	C	CLS	28	1C	CRSR DER
13	D	RETURN	29	1D	CRSR IZQ
14	E	-	30	1E	CRSR ARR
15	F	-	31	1F	CRSR ABA

Los codigos de 1 a 26 se pueden obtener tambien mediante la combinacion de la tecla CTRL y uno de los caracteres de la A a la Z



LOS MEJORES DE INPUT MSX

PUESTO	TITULO	PORCENTAJE
1.°	Knight Lore	21,8 %
2.°	H.E.R.O.	18,1 %
3.°	Soccer	14,3 %
4.°	Profanation	10,1 %
5.°	Alien 8	6,8 %
6.°	Yie ar kung fu	6,2 %
7.°	Hyper rally	6,2 %
8.°	Road Fighter	5,6 %
9.°	Ghostbusters	5,6 %
10.°	The way of the tiger	5,3 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Septiembre de 1986.

HACE YA ALGUNOS AÑOS

Hace ya unos cuantos que comenzó la fiebre de los videos juegos. Muchos de vosotros recordaréis aquellos días. Primero aparecieron, en las máquinas de algunos bares y cafeterías, unos programas para jugar al tenis en los que las raquetas eran dos simples rectángulos y la pelota un cuadrado. Luego apareció otro programa que llegó a ser de los más famosos y que se llamaba algo así como El rompe-ladrillos. Un poco después hicieron su aparición dos programas, ya más elaborados, con gráficos todavía primitivos pero desde luego más interesantes que los simples rectángulos. Nos estamos refiriendo al famosísimo **Comecocos** y al programa objeto de este comentario: **Galaxian**.

Galaxian es el típico juego de marcianitos. Una flota de naves aparece suspendida en la parte superior de la pantalla. En la parte inferior está la nave del jugador, que puede moverse horizontalmente y lanzar disparos láser hacia arriba. Poco a poco, algunas naves de la flota enemiga abandonan la formación y se lanzan, a toda velocidad y disparando, contra la nave del jugador. Los primeros ataques son fáciles de neutralizar, pero a medida que avanza el juego y aparecen nuevas flotas enemigas, para sustituir a las que ha destruido el jugador, este se da cuenta de que su propia destrucción es sólo cuestión de tiempo, porque el número de naves que atacan es cada vez mayor y además se mueven cada vez más deprisa. Lo único que puede hacer el sufrido jugador es intentar resistir el mayor tiempo posible y acumular puntos con los que batir el *record* del dia.

La versión **MSX** de este **Galaxian** es de la firma **Nanco**, autora de la versión original. Y hay que reconocer que lo han conseguido; la versión es perfecta. Los gráficos son idénticos, el movimiento es el mismo, rápido, suave, progresivo. Hasta los sonidos de vuelo de las nave, de los disparos y de las explosiones son exactamente los mismos de la versión original. Nos quitamos el sombrero.

Aunque el juego en sí está un poco anticuado, los que no jugaron con él en su dia o los que quieran recordar los buenos viejos tiempos, no van a encontrar una versión mejor.

DATOS GENERALES

TITULO Galaxian

FABRICANTE Nanco (Philips)

CLASE DE PROGRAMA Marcianitos

FORMATO Cartucho ROM

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	4
INTERES	9
GRAFICOS	6
COLOR	6
SONIDO	9
TOTAL	34





****

JACK EL TERRIBLE

Jack es un joven infante (probablemente todavía no llegue al año de edad), que vive con su familia en una curiosa casa de varias habitaciones. La casa es curiosa porque además de la familia de **Jack**, la habitan una serie de fantasmas, de distintas formas y tamaños, que no hacen más que dar vueltas y vueltas por las habitaciones. **Jack**, controlado por el *joystick* o el teclado, tiene que desplazarse por la casa dando saltos, subiéndose a todos los sitios a los que no debe subirse y haciendo el mayor número de trastadas posibles, al mismo tiempo que evita los encuentros con los distintos miembros de la familia. Al encontrarse con ellos disminuirá su energía. Al agotase esta, **Jack** recibirá una paliza que aparecerá anotada en un marcador.

Lo primero que sorprende de **Jack** es lo buenísimos que son sus gráficos. Las distintas habitaciones de la casa están dibujadas con todo lujo de detalles. Muebles, cuadros, objetos y personajes son una delicia, aunque no se haga mucho uso del color. Recuerdan a los gráficos de algunos excelentes programas tipo **Knight Lore**,





DATOS GENERALES

TITULO Jack The Nipper

FABRICANTE Erbe

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	8
GRAFICOS	10
COLOR	8
SONIDO	8
TOTAL	43





aunque la perspectiva sea distinta, más frontal. En cualquier caso el resultado es excelente.

Jack se puede desplazar en cualquier dirección, pasando de unas habitaciones a otras, y puede saltar. El salto se realiza pulsando la barra de espacios o el botón de disparo del *joystick* y permite que el protagonista se encarame en cualquier mueble y pueda con ello esquivar a los miembros de la familia.

Además de lo bien hechos que están los gráficos (ahí están las fotos de pantalla) **Jack** destaca por el movimiento de los personajes, que resulta suave y perfectamente progresivo. Entre los miembros de la familia hay un simpático perro, que persigue a **Jack** en cuanto le ve.

****

EL EQUIPO DE FONTANEROS

La idea del juego no es nueva; una fábrica o un complejo industrial en el que el elemento fundamental lo constituye una larga y serpenteante tubería que traslada los fluidos de fabricación. En el complejo hay un par de alimañas, concretamente un ratón y una araña, que se divierten cerrando las llaves de paso de la tubería. Y por último los protagonistas, un par de fontaneros (maestro y aprendiz) que tienen que abrir todas las llaves que el ratón ha cerrado, para que siga fluyendo el líquido y pueda continuar la fabricación.

Este es el argumento de **Splash**. El jugador hace las veces de maestro fontanero. Su misión es hacer que trabaje el aprendiz, que, aquí entre nosotros, no es muy aficionado al trabajo. Para motivarle, el maestro tiene que acompañarle hasta donde están las llaves cerradas, andando delante del aprendiz, para que éste, al llegar hasta la llave se decida a abrirla.

Es uno de los aspectos más simpáticos del juego; la aparente indolencia del aprendiz, que una vez que ha abierto una llave se queda ahí, de pie, esperando a que el maestro vuelva a por él y le lleve hasta la siguiente.



DATOS GENERALES

TITULO Splash

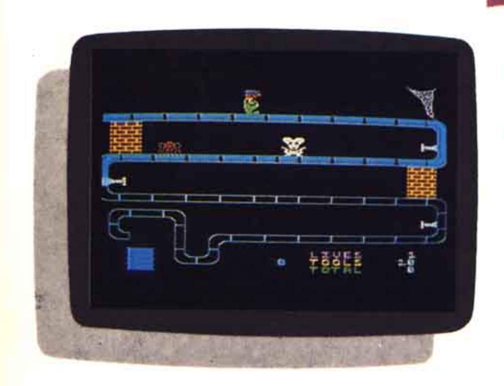
FABRICANTE Artificia Intelligence

CLASE DE PROGRAMA Arcade

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	7
GRAFICOS	9
COLOR	8
SONIDO	8
TOTAL	39



El maestro tiene otra misión importantísima. Consiste en lanzar llaves inglesas contra la araña cada vez que se pone a tiro. Al alcanzarla se consigue que pierda el sentido durante unos minutos, tiempo que se puede aprovechar para abrir tranquilamente alguna de las llaves cerradas.
El juego consta de varias fases que representan distintas zonas de la planta de fabricación. Se accede a la fase siguiente cada vez que la cantidad de líquido que ha llegado a su destino alcanza los 1000 litros, lo que no resulta tarea fácil ni siquiera en la primera fase. El ratón se mueve demasiado deprisa y de una forma totalmente impredecible, por lo que a veces resulta inevitable el encontronazo que supone perder una vida. Da la impresión de que la rutina de movimiento del ratón no ha sido demasiado trabajada.
El programa es divertido, más o menos original, con buenos gráficos (especialmente divertido el del aprendiz), pero tiene un pequeño problema en cuanto los movimientos del ratón y de los personajes, que resultan un tanto incontrolables. Durante las primeras partidas esto



llega a resultar desconcertante, pero poco a poco se coge el truco al *joystick* o al teclado y se consigue un cierto dominio del juego. La musica de fondo es muy buena. En fin, un programa divertido y sin complicaciones, que sin ser ninguna maravilla, merece un puesto en cualquier biblioteca de programas **MSX**.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

OPERACIONES ESPECIALES

Lothlorien, la firma creadora del programa, nos ofrece una mezcla de juego de estrategia militar y juego de aventura conversacional bastante afortunada. La historia es la de un jefe de operaciones especiales, el jugador, que tiene que llevar a cabo una misión. El primer punto curioso es que la misión, que puede elegirse entre 7 diferentes, no será nunca igual que otras misiones. El juego incluye una serie de elementos aleatorios que cambian desde los escenarios hasta los personajes que intervienen cada vez que se juega. Cada misión se puede jugar a lo largo de varios dias, pues el programa preveé la posibilidad de guardar la misión en *cassette*, en un instante cualquiera del desarrollo de la misma, para recuperarla al cabo del tiempo cogiendo el hilo de la historia donde lo habíamos dejado.
Lo primero que tiene que hacer el jefe al principio de la misión es escoger el equipo de hombres (y mujeres) que van a acompañarle. Para ello dispone de un menú con múltiples personajes, cada uno especialista en una actividad determinada (explosivos, cartografía, falsificaciones, etc). Sólo cuatro de los aspirantes serán seleccionados por el jefe. La selección tendrá lugar tras una serie de entrevistas con los candidatos, siempre que haya tiempo

DATOS GENERALES

TITULO Special Operations

FABRICANTE Lothlorien

CLASE DE PROGRAMA
Juego de estrategia

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	10
INTERES	8
GRAFICOS	7
COLOR	8
SONIDO	6
TOTAL	39



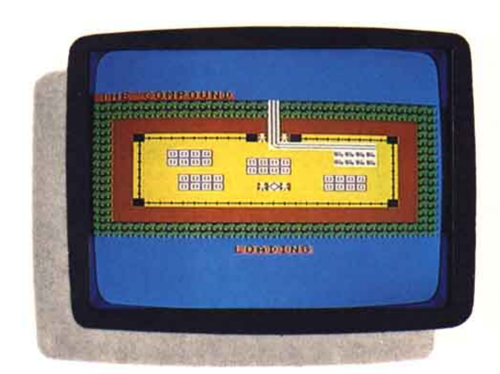




para llevarlas a cabo. Toda la misión se desarrolla alrededor de 3 escenarios principales (aunque el juego incluye 18 mapas distintos) que representan un bosque, un campamento enemigo y un laberinto subterráneo. Estos escenarios, que aparecen dibujados en la pantalla a lo largo del juego, son el lugar en el que se desarrollan combates con enemigos, búsqueda de claves, exploración de cuevas y un sinfín de actividades más.

El juego incluye elementos característicos de las aventuras conversacionales. Por ejemplo se pueden recoger, transportar y dejar objetos, en cualquiera de los escenarios de la misión. La escuadra se moverá a traves de una serie de órdenes de desplazamiento del tipo NE (ir al noreste) y en general todas las acciones se determinarán empleando una especie de lenguaje (tecleando palabras clave) que incluye órdenes como NO, no hacer nada, ST, informe del estado de la escuadra, AT, atacar a los soldados enemigos, etc. A lo largo de la misión el jugador-jefe tiene que emplear adecuadamente las órdenes y tiene que aprovechar en el momento adecuado de las habilidades especiales de los miembros del equipo.

Este argumento y esta complejidad de manejo y funcionamiento, encierran un juego entretenido, no demasiado resuelto en algunos aspectos (por ejemplo en el aspecto conversacional, en el que sólo se pueden introducir determinadas palabras clave y que no tiene una presentación demasiado buena) pero bastante original, con bastantes elementos nuevos y con el enorme aliciente que supone poder entrar en una aventura nueva cada vez que se



juega. Especialmente divertidos son los combates en el bosque, contra las escuadras de vigilantes enemigos. En ellos el jefe decide con qué enemigo combatirá cada miembro de la escuadra. Una vez hecho esto, ordena los movimientos de cada uno de los miembros, intentando que tengan un buen ángulo de tiro y que, al mismo tiempo, estén a cubierto. Cuando ha terminado de moverlos, se inicia el combate en el que cada miembro de la escuadra y también cada uno de los soldados enemigos, efectúa un solo disparo. Según lo bien situada y lo bien cubierta que esté la escuadra, el resultado será mejor o peor.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

CRUDO Y PETRODOLARES

Oils Well te convierte en capataz de una explotacion petrolífera, y te coloca como encargado del manejo de la máquina perforadora. Esta está constituida por una serie de tubos, empalmados unos a continuación de otros. Al extremo de los tubos hay un cabezal o taladro. Tu, como capataz tienes que dirigir el taladro, conduciéndolo a través de una serie de galerías subterráneas en las que encontrarás unos pequeños rectángulos con petroleo, que podrás comerte, al viejo estilo de los juegos de comecocos. Cuando te hayas hecho con todos los rectángulos de una pantalla (cada pantalla recibe el nombre de nivel), pasarás al nivel siguiente. En éste, el entramado de galerías a explorar será más complejo, con más vueltas, recovecos, etc.



Unas cuantas clases de monstruos, de diversas formas y colores, intentarán comerse tus tuberías. De ellos, el más peligroso es «la mina», una carga explosiva que no hay forma de neutralizar y ante la que sólo queda la huida, que no siempre es posible. El resto de los monstruos son comestibles. Puedes comértelos o destruirlos si los alcanzas con el cabezal de la perforadora. Estos monstruos aparecerán por la izquierda y por la derecha, de forma aleatoria, a través de las galerías y a distintos niveles de profundidad. Si alcanzan alguna zona de la tubería antes de que hayas podido



neutralizarlos con el cabezal, perderás una vida.
Lo mejor del juego es el sistema de guía de la perforadora. Con el *joystick* o el teclado, como prefieras, harás avanzar el cabezal a través de las galerías de la forma habitual, lo que ya no es tan habitual es la forma de retroceder. Pulsando el boton de disparo o la barra de espacios, todas las tuberías retroceden a gran velocidad hacia la caseta de exploracion, en la parte de arriba de la pantalla. En las primeras pantallas, esta velocidad de retroceso es suficiente para hacer frente con rapidez a cualquiera de los monstruos, que pueden aparecer por la galeria superior mientras el cabezal de la taladradora está en la inferior. El problema aparece al pasar a niveles superiores, en los que ademas de aparecer más monstruos moviéndose más deprisa, las galerías son más complejas.

DATOS GENERALES

TITULO Oils Well

FABRICANTE Aackosoft

CLASE DE PROGRAMA Arcade

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)	
ORIGINALIDAD	10
INTERES	9
GRAFICOS	9
COLOR	10
SONIDO	7
TOTAL	45



El capataz inicia la jornada con tres vidas y dispone de un tiempo limitado para acabar con cada nivel. El programa es fundamentalmente original, pero además tiene unos gráficos estupendos y un movimiento superconseguido. Al menos nosotros nos lo hemos pasado en grande dirigiendo el taladro y sobre todo haciéndolo retroceder a toda velocidad.

****

LA RANA URBANA

Hopper es una versión del clásico juego de la ranita que tiene que atravesar una concurrida autopista y luego un río, esquivando los coches y camiones que circulan a toda velocidad por los carriles de la autopista y saltando de tronco en tronco en el río hasta llegar a las bahías del otro lado.
Hay que evitar a las peligrosas culebras, nutrias y cocodrilos que intentarán comerse a la rana mientras salta de tronco en tronco y de tortuga en tortuga.
En algunos de los troncos, tomando el sol, hay una rana hembra; si la ayudamos a pasar al otro lado obtendremos un buen montón de puntos extra como premio. Además, si nos comemos las moscas, que aparecen de forma aleatoria en las bahías del otro lado del río, recibiremos unos cuantos puntos más.
Cuando hayamos hecho pasar a todas las ranas a la otra orilla del rio, recibiremos otro montón de puntos y pasaremos a otra pantalla con mayor nivel de dificultad.
Hopper no ofrece ninguna innovación sobre este clásico juego, de ahí su escasa puntuación en originalidad. De todas formas es una versión fiel, bien realizada y que no tiene nada que envidiar a la versión original del juego, aparecida hace ya unos años, y que muchos de vosotros, sin duda, disfrutásteis en su día. Los gráficos, el color, el sonido y los movimientos de los distintos *sprites* que intervienen, son todos ellos una reproducción fiel de la versión original.
El juego en sí es entretenido, hay que reconocer que a estas alturas resulta un tanto anticuado, pero su buena realización le permite competir con algun que otro juego mucho más actual. Hay que considerarlo como un clásico de los juegos de ordenador.

DATOS GENERALES

TITULO Hopper

FABRICANTE Aackosoft

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)	
ORIGINALIDAD	6
INTERES	6
GRAFICOS	7
COLOR	7
SONIDO	6
TOTAL	32





EL ZOCO

Vendo ordenador Sony de 32K, regalo dos cartuchos de juegos. Preguntar por:

Fernando
Tel. 386 30 19
Barcelona

Intercambio ampliación de 16K más 30 juegos por una ampliación de 64K que esté en buen estado. Interesados llamar a:

Manolo
Tel. (954) 63 21 94

Vendo unidad de discos MSX Philips VY 0010 impecable, con programas de tratamiento de textos, base de datos, control de stocks, presupuestos, pedidos, sistema DOS, ensamblador y buenos juegos en diskette. Todo ello por 59.000 ptas.

José Luis
Tel. (93) 870 21 90 (de 20 a 22 horas)

Vendo ordenador HB 55P de Sony, incluyo también ampliación de 16K de RAM y regalo tres cartuchos de juegos.

Fernando Algar Alarcón
Tel. 386 30 19
Barcelona

Vendo ordenador MSX HIT-BIT 75-B de Sony a buen precio. Escribir o llamar:

Almogávares, 17
Tel. 83 01 80
Llagostera (Gerona)

Vendo video-juego CBS, con 4 cartuchos: «Mouse Trap», «Zaxxon», «Miner 2049» y «Donkey Kong». Todo por 20.000 ptas. Al comprar regalo: Máquina Gare - Watch (escoger). Llamen a:

Tel. 254 77 98
Barcelona

Vendo ordenador MSX Philips VG-8010 de 48K de RAM, con 50 programas comerciales. Buen precio.

José Luis Stoduto García
Manuel Blasco, 8, 5ºC
42004 Soria

Vendo/cambio. Desearía intercambiar juegos del MSX. Tengo: Gunfright, Sorcery, Alien 8, Nightshade, Knightlore y 30 programas más. Y desearía vender 2 copiones diferentes, hechos por mi, en código máquina. Y desaría tener en MSX, Rambo y Sobrewulf, Profanation, Turmoil, Misión Espacial, The way of the tiger, etc.

Tel. 20 75 56
Castellón

Intercambio programas comerciales MSX en cinta y en disco de 3.5 pulgadas.

Irene Juarros
Garita, 19
Tel. (971) 40 36 59
Palma de Mallorca

Vendo videojuego televisión con deportes: Futbol, Squash, Balonmano, Tenis, Frontón. Con pistola para tiro al plato por la T.V. Al comprar regalo máquina Gare-Watch (escoger). Sólo por 10.000 ptas. Llamen a:

Tel. 254 77 98
Barcelona

Intercambio conocimientos en lenguaje máquina, para crear programas. Busco mapa de memoria. Llamar a:

Francisco Javier
Tel. (91) 200 96 72

Vendo juegos MSX, tales como Ghostbuster y Pitfall II, por sólo 2.500 cada uno. Aparte vendo cassette Load'n'run y Guss en la Atlantida por 500 ptas. los dos. Ruego llamar:

Tel. 254 77 98
Barcelona

Intercambio todo tipo de programas MSX. Me interesaría comprar una unidad de discos MSX.

David Ibarra
Unbe-Mendi 16 Laukiniz
Tel. (94) 453 06 94
48080 Bilbao

Somos un grupo libre que nòs gustaría contactar con chicos de toda España y del extranjero, para intercambiar juegos del C-64 y del MSX.

Santi Barbará Molina
Jocs Florals s/n Lª A 2º
Badalona (Barcelona)

Contactar con chicos/as de toda España y en especial de Guadalajara para intercambiar programas. Escribir a:

Enrique Alda Ruiz
Avda. de Castilla, 18 F, 3ºD
19002 Guadalajara

Cambio ampliación de 16K por una de 64K pagando diferencia 8.000 pts. Interesados escribir a:

Ois Santiago
Cordonería 3 y 5, 2º centro
Tel. (981) 21 35 53
15001 La Coruña

Intercambio juegos en cinta para MSX (Knight Lore, River Raid, Chess, 86, etc...) por otros (Zaxxon, Hero, etc...). Llamar o escribir a:

Xavier Martinez
Emérita Augusta 10 esc. B 3 2
Tel 339 86 22
08028 Barcelona

Vendo SVI-328 + cassette SVI 904 + joystick quickshot I + 16 programas juegos y utilidades (Ninja, Turboat, ...) + manuales y revistas. **Todo por 46.000** ptas.

Sigfrido Blay Sanchez
Avda. Pérez Galdos, 97, pt 18
Tel 326 49 16
46018 Valencia

Intercambio todo tipo de programas en cinta o disco de 3 1/2. Interesados llamar o escribir a:

Joan Taulats Vila
Ausias March, 65, 3 1
Tel. (93) 245 99 84
08010 Barcelona

TU PUEDES SER EL AFORTUNADO

Si quieres participar en el concurso promovido por Erbe Software, recorta y pega esta solapa que te permitirá completar la página de acceso al sorteo.



Desearía intercambiar programas en MSX. Poseo: J. Set Willy II, Gunfright, Knight Lore, Nightshade, Le Mans, Sorcery, hasta 40 títulos más. Me interesaría: Profanation, The Way of the Tiger o Manic Miner. Escribir señalando vuestro teléfono a:

Xavier Giralt Llopart
Avda. José Mª Valls, 17
Castellar del Vallés
Barcelona

Vendo ordenador Sony HB-75P, 80K, en perfecto estado, con cables, manuales y unos 60 programas (juegos y de aplicación) por 45.000 ptas. Opcionalmente vendo grabadora Sony Bitcorder SDC-500. Llamar o escribir a:

Salvador Roura
Pasaje Emsesa 7, pral. 2ª
Tel. (973) 53 07 81 de 4 a 6 tarde
25200 Cervera (Lérida)

Cambio cartuchos Konami (Hyper-Sports 1, Track-Field 2, Soccer) por otros de la misma marca, iguales condiciones.

Jordi Mayá Meseguè
San Jose, 14
Tel. (93) 769 06 40
Sabados 6 a 9 tarde
Calella (Barcelona)

Vendo ordenador Philips VG8020 MSX de 80K RAM, muy nuevo, por cambio de sistema. Buen precio.

Alberto Martínez Correro
Palencia, 27-29, esc. A, 8º, 3ª
08027 Barcelona

Cambio cartucho Soccer de Konami, Hyper Sports 3 o Yie Ar Kung-Fu 2 por cualquier otro. También poseo mapas y pokes de Gunfright, Knigth Lore, Alien 8 y Nigth Shade. Escribir o llamar a:

Jordi Codina
Bilbao 11, 4º 3ª
Tel. (93) 873 41 35
Manresa (Barcelona)

Cambio buenos juegos (Soccer, King's Valley, Circus Charlie...) por Knight Lore. Intentad ser de Barcelona capital. Llamar a:

Tel. (93) 322 64 33
A partir de las 18 horas

Intercambio juegos para MSX.

Manuel Piñeiro
Gimbernat, 19, Atc. 3
Tel. (93) 593 60 63
Mollet del Vallés
Barcelona

Desearía comprar cartucho de ampliación de 16K. Precio a convenir. Interesados llamar a:

Ismael
Tel. (93) 385 04 94
Sta Coloma de Gramanet
Barcelona

Cambio cartucho de ampliación 64K de la marca Sony o SVI por una serie de juegos entre ellos Alien-8, H.E.R.O., Manic Miner, Maciac, River Raid, etc. más 4.000 ptas. en metálico. Estaría también dispuesto a comprarlo. Interesados escribir:

Apartado 342
Vitoria (Alava)

Vendo por cambio de ordenador, Hit Bit Sony 101 P, poco uso, cables de conexión, manuales en español y un cartucho y una cinta, todo sólo por 35.000 ptas.

Angel Encabo Gómez
C.S. Ant.Mª. Claret 357, 2º 2ª esc.A
Tel. (93) 349 09 44 de 8 a 10 noche
08027 Barcelona

Cambiaría contabilidad doméstica (cas) y Flight Path 737 (sim. vuelo cas.) por Zaxxon o Soccer. Compraría unidad diskette Sony o SVC.

Martín Piqueras Caro
Selva, 25 (Restaurant)
Tel. 388 19 08
Badalona (Barcelona)

Cambio/Vendo: Peetan, Mr. Ching, Soccer, Gunfright, King's Valley, Circus Charly, Ping-Pong, Hyper-Rally, Sky.

Tel. (93) 321 48 72

Deseo contactar con usuarios del MSX para poder cambiar programas y opiniones, etc... a ser posible de la provincia de Cádiz. Llamar o escribir a:

Manuel López Jarillo
Avda. León de Carranza 2, 3ºB
Edificio El Cuco
Tel. (956) 31 43 99
Jerez de la Frontera (Cádiz)

Intercambio programas comerciales, tengo gran cantidad de programas de cartucho y cinta.

Francisco Jimenez Zambrana
Tesino, 14, Bª Las Portadas
Dos Hermanas (Sevilla)

Cambio 32 juegos de revistas y 32 comerciales por un cartucho de ampliación de memoria de 64K. Se podrá llegar a un acuerdo de compra también.

Jesús Contreras Luna
Angelita Capdevilla, 5, 3º izda.
Tel. (927) 22 38 19
10002 Cáceres

Cambio o vendo juegos entre otros Pitfall II, Hiper Sports I, Sky Jaguar, River Raid, Antartic Adventure, Galaxian y Monkey Academy. Escribir o llamar a:

Juan Enrique Camps
Poeta Mas y Ros, 124, 1,1
Tel. (96) 372 95 64
36022 Valencia

Cambio juegos en cinta Beamrider o Pitfall II (originales) por las cassettes originales de Knight Lore o Hero. Llamar a:

Alberto Salek
Caldas de Reyes, bloque E-15, 1ºB
Tel. (986) 23 21 80
Vigo (Pontevedra)

Cambio ZX-Spectrum 48K+Interface Kempston+Joystick+500 programas por ordenador MSX de 64K RAM. Interesados/as escribir a:

Guillermo Rodriguez Bey
Juan Van Halen, 16, 1º dcha.
Tel. (956) 89 44 54 de 4 H. en adelante
San Fernando (Cádiz)

Cambio juegos MSX, poseo Knigth Lore, Pitfall II, Ghostbusters, Decathlon, Le Mans, Piramid Warp, Jet Set Willy, etc. Cambio cartucho Castle Combat por otro. Escribir a:

Tico Moreno
Cuenca, bloque L, bajos 3ª
Montcada (Barcelona)

Cambio juegos (Hero, Kinght Lore, Ghostbusters, Gunfright, Buck Rogers, Zaxxon, Decathlon, Sorcery, etc.) por The way of the tiger, Profanation, Nigthshade, Alien 8 o Modes of Yesod.

Jorge Fábrega
Tel. (973) 35 14 74

Vendo ordenador nuevo marca Canon V-20.

Tel. (91) 303 32 78
A partir de las 22 h.

Vendo cartuchos Track Field I, Juno First y Mouser. Todos por 8.000 ptas. y la unidad por 3.000 ptas. Pedidos contrareembolso.

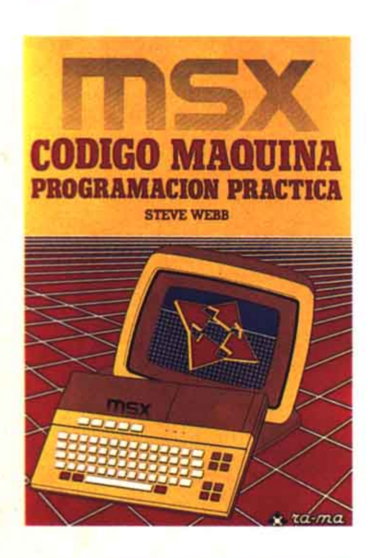
Tel. (956) 30 34 95

Compro Hit-Bit Sony con más de 32K por 48.000 ptas, incluyendo monitor MSX de fósforo verde, y demás accesorios del ordenador (cables, etc.). Se estudiarán otros precios. Llamar a:

Germán Gozalo
Tel. (986) 29 27 22
Vigo (Pontevedra)

Vendo ordenador Philips VG 8000, un cartucho de ampliación de 16K, 13 cintas de programas. Precio a convenir. Llamar a:

Jose Antonio
Tel. (93) 337 94 79



MSX CODIGO MAQUINA: PROGRAMACION PRACTICA

Autor: Steve Webb
Editor: Rama
Páginas: 128
Precio: 1.200

Con este libro se presenta una breve introducción a la programación en código máquina del microprocesador incorporado en los ordenadores **MSX**, el **Z80**.

Tras una breve introducción en la que se habla del estándar **MSX**, se explica de forma sencilla y básica qué es el código máquina, y cuales son los equivalentes en código máquina de las operaciones más importantes del BASIC, haciendo así de puente entre ambas formas de programación, y ayudándonos de paso a entender cómo ejecuta el intérprete de BASIC de nuestro ordenador las sentencias de este lenguaje.

Después se trata el tema del almacenamiento en memoria de los códigos de operación, aprovechándose aquí para la introducción del código hexadecimal y suministrándose un pequeño programa para escribir el código máquina que compone el programa en memoria.

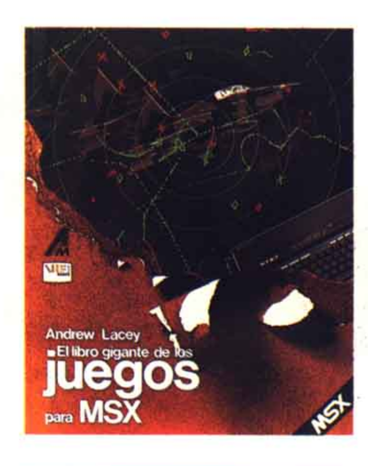
Se incluyen asimismo una explicación de la gestión de pantalla y de los gráficos, y algunas rutinas de utilidad del sistema operativo grabado en ROM de nuestro ordenador.

Como ejemplo concreto de programación se desarrolla un juego llamado «Invasor del Espacio», pasándose del esquema general de bloques, al código máquina que compone cada uno de ellos.

Por último se ofrecen una serie de rutinas de utilidad que el lector podrá usar como pequeñas utilidades incluidas en su programa, evitándose tener que construirlas él mismo. Tenemos rutinas para mover el texto por la pantalla, y efectos sonoros tales como una bomba y un láser.

En los apéndices se incluyen una lista completa de los códigos de operación del **Z80**, una tabla de conversión de binario a hexadecimal, una pequeña explicación de código binario y un programa diseñador de caracteres.

En resumen, un libro interesante para los principiantes en este tipo de programación. Se echa de menos sin embargo una explicación más ordenada y completa del ensamblador del microprocesador **Z80**.



EL LIBRO GIGANTE DE LOS JUEGOS PARA MSX

Autor: Andrew Lacey
Editor: Anaya
Páginas: 302
Precio: 1.590

La gran mayoría de los libros dedicados a ofrecer programas de juegos se limitan a suministrar los listados acompañándolos de un sucinto comentario, muchas veces poco o nada ilustrativo.

No es éste el caso de este libro. En él aparecen 27 interesantes programas de juegos para nuestro ordenador **MSX**. Además, aparecen una serie de innovaciones frente a lo que vienen siendo normal en este tipo de libros.

Es muy común que, tras habernos pasado toda una tarde introduciendo un programa en el ordenador, nos encontremos con que, al ejecutarlo, o hemos cometido un error sintáctico, o el programa no funciona bien por haber escrito mal una sentencia.

Esto se evita, en parte, con el programa verificador incluido en el libro. Este programa produce un código numérico para cada línea de programa que puede compararse con el código numérico que aparece al final del listado. Con ello puede hacerse una verificación rápida de si el programa se escribió correctamente. Además se puede localizar rápidamente en qué línea se cometió el error. Esta verificación, sin embargo, no es completa, puesto que puede darse la desgraciada casualidad de que dos errores en una misma línea sean tales que su suma no afecte al código numérico.

Al comienzo de cada juego hay un comentario informal sobre su cometido. Los listados están estructurados en bloques, discutiéndose brevemente la misión de cada uno de ellos.

Hay que resaltar muy especialmente que cada programa contiene un listado de las variables que se usan y de su función, asi como unas sugerencias para los lectores que deseen modificar el programa.

SISTEMAS EXPERTOS. CONCEPTOS Y EJEMPLOS

Autor: J.L. Alty y M.J. Coombs
Editor: Díaz de Santos
Páginas: 218
Precio: 2.120 ptas.

Para aquellos que están muy interesados en el mundo de la **Inteligencia Artificial** y los **Sistemas Expertos**, a condición de que ya posean un buen nivel de conocimientos y muchas ganas de aprender, recomendamos especialmente un libro de reciente aparición que promete satisfacer la curiosidad de los más avezados devoradores de bibliografía. Advertimos que su contenido no es de aplicación concreta en ningún ordenador específico, sino que se trata de una descripción exhaustiva de los principios generales de los **S.E.** a un nivel profundo.

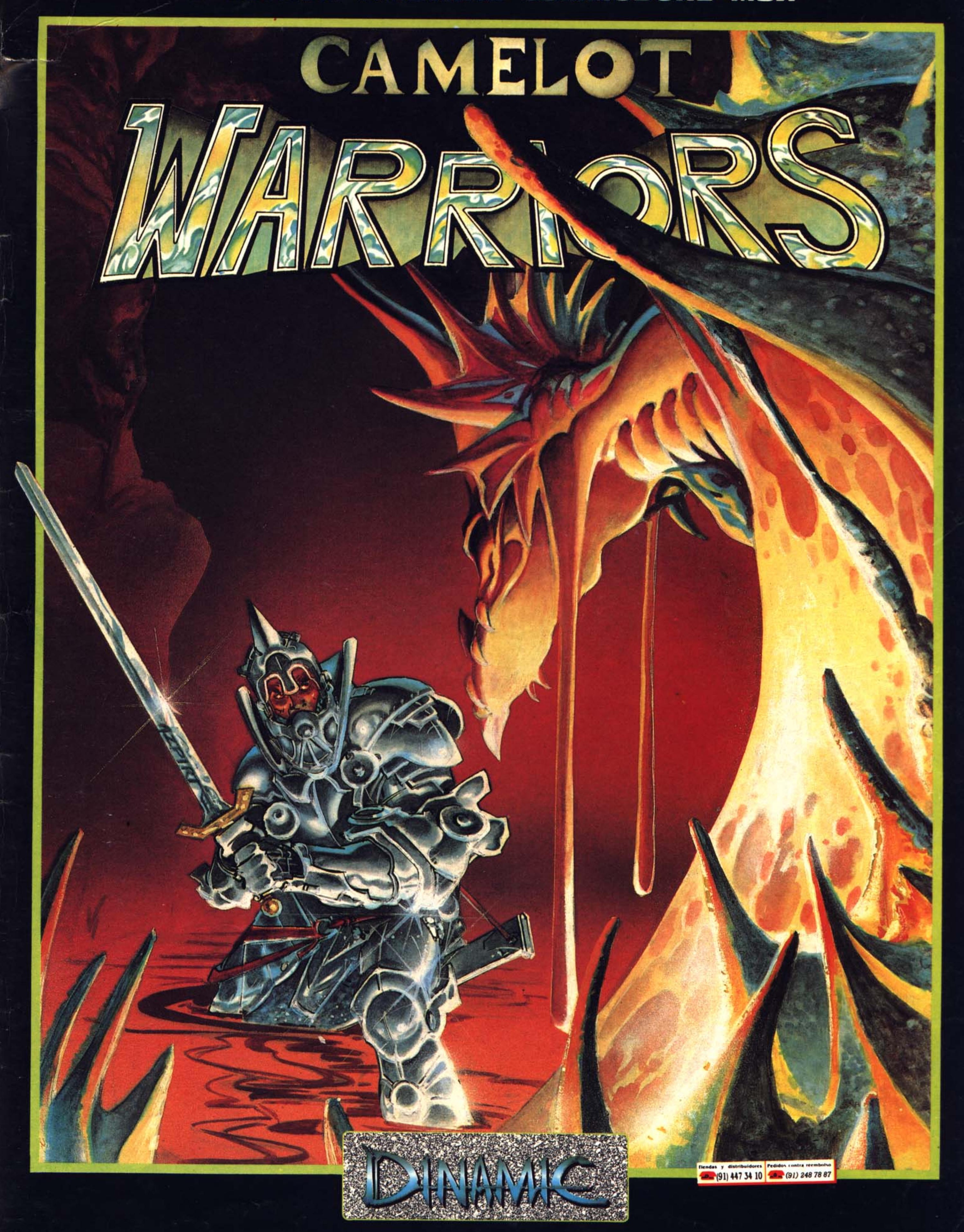
Los autores comienzan la obra comparando el proceso de datos tradicional con los **Sistemas Expertos**, para pasar seguidamente a describir con detalle los métodos de representación y control, la «inferencia lógica», el cálculo de predicados, y su aplicación para la resolución de problemas.

Nos hablan también del ya tópico sistema de «factorización» para la reducción del tiempo empleado en buscar un dato en una tabla.

Desgraciadamente, una de las partes más interesantes se refiere a diversos sistemas existentes en el mercado, como el famoso «PROSPECTOR», que no están al alcance de nuestro bolsillo y mucho menos de nuestro «micro» (y de los que no son tan «micros»). No obstante, bien es cierto que no es posible felicitarnos por nuestros conocimientos sobre **Inteligencia Artificial** sin haber oído hablar de dichos sistemas.

La obra termina, como no, con una panorámica de los nuevos desarrollos en materia de **Sistemas Expertos** muy actualizada (aunque sepamos que en un par de años se quedará obsoleta),




SPECTRUM · AMSTRAD · COMMODORE · MSX
CAMELOT
WARRIORS
DINAMIC
Tiendas y distribuidores
(91) 447 34 10
Pedidos contra reembolso
(91) 248 78 87